REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-

0477

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 14 May 2001	3. REPORT TYPE AND DATES COVERED Final Technical Report (15 May 1998 - 14 May 2001)

4. TITLE AND SUBTITLE
Discrete Manufacturing Process Design Optimization: Theory and Application

5. FUNDING NUMBERS
F49620-98-1-0432

6. AUTHOR(S)
Richard Nance
Subcontract to: Sheldon H. Jacobson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
VPI&SU, 340 Burruss Hall, Blacksburg, VA 24061

8. PERFORMING ORGANIZATION REPORT NUMBER
#1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR
801 North Randolph Street, Room 732
Arlington, VA 22203-1977

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR

11. SUPPLEMENTARY NOTES
None

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

12b. DISTRIBUTION CODE

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF TRANSMITTAL DTIC. THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLIC RELEASE LAW AFR 190-12. DISTRIBUTION IS UNLIMITED.

13. ABSTRACT *(Maximum 200 words)*

Discrete manufacturing process design optimization (DMPDO) is a problem of significant importance and interest to the Air Force. Moreover, the complexity of parts that must be manufactured for airplane engines and other units makes this problem extremely difficult. This report summarizes the research that has been conducted in developing the generalized hill climbing (GHC) algorithm framework for discrete optimization problems in general, and the DMPDO problems in particular. Convergence conditions for GHC algorithms have been developed. Ordinal hill climbing (OHC) algorithms were introduced to exploit the efficiency of ordinal optimization and the effectiveness of GHC algorithms to obtain a new class of discrete optimization problem algorithms. The relationship between OHC algorithms and genetic algorithms was also studied. Multiple sequence optimization using GHC algorithms were introduced to optimize across sets of DMPDO sequences. Simultaneous GHC algorithms were introduced as a generalization to optimize across a set of related discrete optimization problems. On-going interactions with researchers at the Materials Process Design Branch of the AFRL at WPAFB and at Austral Engineering and Software, Inc. has resulted in these algorithms being transitioned for application into commercial software tools designed to solve various DMPDO problems of interest to the Air Force.

14. SUBJECT TERMS
Manufacturing Process Design Optimization, Discrete Optimization, Heuristics

15. NUMBER OF PAGES
45

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

FINAL TECHNICAL REPORT

Submitted to

Dr. Neal D. Glassman
AFOSR/NM
801 North Randolph Street, Room 732
Arlington, VA 22203-1977
(703) 696-8431
neal.glassman@afosr.af.mil

Principal Investigator:

Sheldon H. Jacobson
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2906
(217) 244-7275 (office)
(217) 244-6534 (fax)
shj@uiuc.edu
www.staff.uiuc.edu/~shj/shj.html

Grant Number: F49620-98-1-0432 (AASERT)

Table of Contents

EXECUTIVE SUMMARY

The research conducted under this grant focused on the development and analysis of the generalized hill climbing algorithm framework as a tool for address intractable discrete optimization problems. The major technical accomplishments achieved include

i) the introduction and development of a new algorithm classification scheme for generalized hill climbing algorithms,

ii) the formulation of new necessary and sufficient conditions for generalized hill climbing algorithms,

iii) the introduction and analysis of ordinal hill climbing algorithms as a new class of generalized hill climbing algorithms,

iv) the introduction and analysis of multiple sequence optimization using generalized hill climbing algorithms as a new class of generalized hill climbing algorithms for optimizing over sets of manufacturing process design sequences simultaneously.

v) the introduction and analysis of simultaneous generalized hill climbing algorithms as a new class of algorithms for optimizing over sets of related discrete optimization problems.

All the accomplishments are documented in several archival journal articles. In addition, the results have been presented at national and international conferences.

Two Ph.D. dissertations were completed during the period of the grant. Dr. Kelly Ann Sullivan successfully defended and submitted her Ph.D. dissertation *On the Convergence of Generalized Hill Climbing Algorithms* in May 1999. Dr. Diane E. Vaughan successfully defended and submitted her Ph.D. dissertation *Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problem* in August 2000. In addition, Mr. Derek E. Armstrong is in the final stages of his dissertation research and it is anticipated that he will defend his Ph.D. dissertation within the next twelve months.

1. Generalized Hill Climbing Algorithms

Generalized hill climbing algorithms (Jacobson et al. 1998) provide a well-defined framework to model algorithms for intractable discrete optimization problems. Generalized hill climbing algorithms allow inferior solutions to be visited enroute to optimal solutions. This hill climbing capability is the basis for the search strategy's name. To formally describe the generalized hill climbing algorithm framework, several definitions are needed.

For a discrete optimization problem, define the *solution space*, Ω, as a finite set of all possible solutions. Define an *objective function* $f: \Omega \rightarrow [0,+\infty]$ that assigns a non-negative value to each element of the solution space. An important component of GHC algorithms is the *neighborhood function*, $\eta: \Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subseteq \Omega$ for all $\omega \in \Omega$. Solutions in a neighborhood are generated *uniformly* at each iteration of a GHC algorithm execution if, for all $\omega \in \Omega$, with $\omega' \in \eta(\omega)$,

$$P\{\omega' \text{ is selected as the neighbor of } \omega \text{ at a given iteration of a GHC algorithm}\} = 1 / |\eta(\omega)|.$$

Unless otherwise stated, assume that the neighbors are generated uniformly at each iteration of a GHC algorithm. The GHC algorithm is described in pseudo-code form (Jacobson et al. 1998):

```
Set the outer loop counter bound K and the inner loop counter bounds N(k), k = 1,2,...,K
Define a set of hill climbing (random) variables $R_k: \Omega \times \Omega \rightarrow \Re \cup \{-\infty,+\infty\}$, k = 1,2,...K
Set the iteration indices    i = 0, k = n = 1
Select an initial solution $\omega(0) \in \Omega$
*Repeat while* $k \leq K$
  *Repeat while* $n \leq N(k)$
    Generate a solution $\omega \in \eta(\omega(i))$ and calculate $\delta(\omega(i),\omega) = f(\omega) - f(\omega(i))$
    If $R_k(\omega(i),\omega) \geq \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega$          (accept improving or hill climbing moves)
    If $R_k(\omega(i),\omega) < \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega(i)$                    (reject hill climbing moves)
    $n \leftarrow n+1$, $i \leftarrow i+1$
  *Until* $n = N(k)$
  $n \leftarrow 1$, $k \leftarrow k+1$
*Until* $k = K$
```

Note that the outer and inner loop bounds, K and N(k), k = 1,2,...,K, respectively, may all be fixed, or K can be fixed and the N(k), k = 1,2,...,K, are random variables whose values are determined by the solution at the end of each set of inner loop iterations satisfying some property (e.g., the solution is a local optima). Moreover, assume that the hill climbing random variables have finite means and finite variances for all k and for all possible pairs of elements in the solution space (i.e., $E[R_k(\omega(i),\omega)] < +\infty$ and $\mathrm{Var}[R_k(\omega(i),\omega)] < +\infty$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, and for all $k = 1,2,\ldots,K$, $i = 1,2,\ldots,I = \sum_{k=1}^{K} N(k)$).

The neighborhood function establishes relationships between the solutions in the solution space, hence allows the solution space to be traversed or searched by moving between solutions. To ensure that the solution space is not fragmented, assume that all the solutions in the solution space (with neighborhood function η) are *reachable;* that is, for all $\omega',\omega'' \in \Omega$, there exists a set of solutions $\omega_1, \omega_2, \ldots, \omega_m \in \Omega$ such that $\omega_r \in \eta(\omega_{r-1})$, $r = 1, 2, \ldots, m+1$, where $\omega' \equiv \omega_0$ and $\omega'' \equiv \omega_{m+1}$. Note that if all solutions in a solution space are reachable, then the solution space (with neighborhood function η) is said to be reachable. The goal is to identify a globally optimal solution ω^* (i.e., $f(\omega^*) \leq f(\omega)$ for all $\omega \in \Omega$).

2. Necessary and Sufficient Convergence Conditions

New necessary and sufficient conditions for the convergence of generalized hill climbing algorithms were obtained during this grant. Convergence conditions provide one way to assess the value and effectiveness of an algorithm. These results are reported in Sullivan (1999) and Sullivan and Jacobson (2001). To describe these conditions, several additional definitions are needed.

The objective function, f, and the neighborhood function, η, allow the solution space, Ω, to be decomposed into three mutually exclusive and exhaustive sets:

- a set of *G-local optima*, G	(the set of global optima),
- a set of *L-local optima*, $L \equiv L(\eta)$	(the set of local optima that are not G-local optima),
- a set of *hill solutions*, H.	

Therefore $G \cup L$ are the set of local optima in Ω associated with neighborhood function η, where by definition, $\Omega = G \cup L \cup H$ with $G \cap L = \varnothing$, $G \cap H = \varnothing$, and $L \cap H = \varnothing$. Note that by definition, for all $\omega \in G$, $\eta(\omega) \cap L = \varnothing$, and for all $\omega \in L$, $\eta(\omega) \cap G = \varnothing$ (i.e., a G-local optimum and a L-local optimum cannot be neighbors of each other).

GHC algorithms can be viewed as sampling procedures over the solution space Ω. The key distinction between different GHC algorithms is in *how* the sampling is performed. For example, Monte Carlo search produces unbiased samples (with replacement) from the solution space, while simulated annealing produces biased samples, guided by the neighborhood function, the objective function, and the temperature parameter. More specifically, simulated annealing can be described as a GHC algorithm by setting $R_k(\omega(i),\omega) = -t_k \ln(\nu_i)$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, $k = 1,2,\ldots,K$, where t_k is the temperature parameter (hence, defines a cooling schedule as $t_k \rightarrow 0$) and $\{\nu_i\}$ are independent and identically distributed U(0,1) random variables (see Johnson and Jacobson 2001a,b for a complete discussion on and description of how these algorithms fit into the GHC algorithm framework).

To ensure that the neighborhood function does not fragment the solution space (i.e., decompose the solution space into mutually exclusive subsets of solutions that are not reachable), a neighborhood function η on the solution space Ω is said to have the *local search property* if when local search (i.e., $R_k(\omega',\omega'') \equiv 0$ for all $\omega' \in \Omega$, $\omega'' \in \eta(\omega')$, $k = 1,2,\ldots,K$) is applied with neighborhood function η to the solution space Ω, then for all $\omega_H \in H$, there exists $\omega_{G \cup L} \in G \cup L$ such that the local search algorithm will terminate in $\omega_{G \cup L}$, passing only through elements in H. Assume that all neighborhood functions η associated with a GHC algorithm applied to solution space Ω have the local search property. Note that if a solution space (with neighborhood function η) is reachable, then η also possesses the local search property, though the reverse is not necessarily true.

Each execution of a GHC algorithm generates a sequence (sample) of $I = \sum_{k=1}^{K} N(k)$ solutions. In practice, the best solution obtained over the entire GHC algorithm run, not just the final solution, is reported. This allows the algorithm to aggressively traverse the solution space visiting many inferior solutions enroute to a globally optimal solution, while retaining the best solution obtained throughout the entire GHC run. Each sequence of solutions is a function of the initial solution, $\omega(0) \in \Omega$, and two sets of independent and identically distributed U(0,1) random variables,

i) $\{\xi_i\}$, that generate the neighbors, $\omega \in \eta(\omega(i))$ (hence, allow $\delta(\omega(i), \omega) = f(\omega) - f(\omega(i))$ to be computed) at each iteration $i = 1,2, \ldots$,

ii) $\{\nu_i\}$, that generate values for $R_k(\omega(i), \omega)$, when $\delta(\omega(i), \omega) > 0$, to determine whether the generated neighbor is accepted or rejected (i.e., $R_k(\omega(i), \omega) \geq \delta(\omega(i), \omega)$ or $R_k(\omega(i), \omega) < \delta(\omega(i), \omega)$, respectively).

Note also that $\{\xi_i\}$ and $\{\nu_i\}$ are independent. In addition, assume that when comparing different GHC algorithms, the initial solution is given and fixed across all such algorithms. This means that $(\{\xi_i\},\{\nu_i\})$ completely define the probability of each sequence of I solutions generated by a GHC algorithm. More specifically, let $\Sigma \equiv \Sigma_{\omega(0)}$ denote all sequences of I solutions generated by a GHC algorithm, where each of the I solutions is in Ω, $\Im$ denotes the sigma field of events on Σ, and $P \equiv P_{\xi,\nu}$ denote the probability measure. Therefore, $(\Sigma, \Im, P)$ defines a probability space on the sequences of I solutions generated by a GHC algorithm, where this probability space is characterized by the initial solution, $\omega(0)$, and the set of independent and identically distributed U(0,1) random variables, $(\{\xi_i\},\{\nu_i\})$, that determine the sequence of I solutions generated by a GHC algorithm. For simplicity and ease of notation, $\omega(0)$ and $(\{\xi_i\},\{\nu_i\})$ will be suppressed, unless they are needed to avoid ambiguities.

The iterations of a GHC algorithm can be classified as either micro or macro iterations (Sullivan 1999). A *micro iteration* moves the algorithm from the current solution either to an immediate neighbor or back to itself. A *macro iteration* is a set of micro iterations that moves the algorithm from any element of $G \cup L$ to any element of $G \cup L$ (including itself), passing only through elements of H.

Using these definitions, at macro iteration k (fixed), the micro iterations define a homogeneous discrete time Markov chain, with *micro iteration transition matrix*

$$P_m^k = \begin{bmatrix} P_m^k(G,G) & P_m^k(G,L) & P_m^k(G,H) \\ P_m^k(L,G) & P_m^k(L,L) & P_m^k(L,H) \\ P_m^k(H,G) & P_m^k(H,L) & P_m^k(H,H) \end{bmatrix}$$

where $P_m^k(U,V)$, $U,V \in \{G,L,H\}$ are $|U|\times|V|$ matrices representing the micro iteration transition probabilities from the elements in set U to the elements in set V. Note that if the micro iteration transition Markov chain is aperiodic and irreducible, then at macro iteration k, the macro

$$P_M^k = \begin{bmatrix} P_m^k(G,H)(I-P_m^k(H,H))^{-1}P_m^k(H,G)+P_m^k(G,G) & P_m^k(G,H)(I-P_m^k(G,H))^{-1}P_m^k(H,L)+P_m^k(G,L) \\ P_m^k(L,H)(I-P_m^k(H,H))^{-1}P_m^k(H,G)+P_m^k(L,G) & P_m^k(L,H)(I-P_m^k(H,H))^{-1}P_m^k(H,L)+P_m^k(L,L) \end{bmatrix}$$

iterations define an inhomogeneous Markov chain, with *macro iteration transition matrix.*

Without loss of generality, assume that the GHC algorithm run is initialized at a solution in L (i.e., $\omega(0) \in L$), since local search can be applied from any element in Ω, and the local search property holds for the solution space Ω. This places a restriction on the classes of discrete optimization problems that can be studied, since if a local optima cannot be obtained in polynomial time in the size of the problem instance, then initializing the GHC algorithm run in this way may not be feasible (see Jacobson and Solow 1993). In addition, if local search is applied and the local optima obtained is a G-local optima, then the problem is solved, though this may not be known until further algorithm iterations are executed.

To illustrate the micro iteration and macro iteration concepts, from the GHC algorithm pseudo-code, let the outer loops represent macro iterations, while the inner loops represent micro iterations between the macro iterations. Therefore, as the GHC algorithm is now defined, there will be K macro iterations, with N(k) micro iterations associated with macro iteration k = 1, 2, ..., K (i.e., N(k) micro iterations between macro iteration k-1 and k, where the 0th macro iteration is defined as the initialization of the GHC algorithm run at $\omega(0) \in L$). Note that all the solutions visited during the N(k) micro iterations associated with macro iteration k will be in H, and that N(k), k = 1,2,...,K, will be random variables.

To obtain the necessary and sufficient convergence conditions for a GHC algorithm to converge, define Π_ω^k to be the long run stationary distribution of a GHC algorithm being at solution $\omega \in G \cup L$ at macro iteration k. The following lemma provides upper and lower bounds on the sum of the long run stationary distribution of being at a solution in L at macro iteration k.

Lemma 1 (Sullivan and Jacobson 2001): For a GHC algorithm at macro iteration k (fixed), with macro iteration transition matrix P_M^k,

$$\min_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma) / [\min_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma) + \max_{\sigma\in L} \sum_{\omega\in G} P_M^k(\sigma,\omega)]$$
$$\leq \sum_{\omega\in L} \Pi_\omega^k \leq$$
$$\max_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma) / [\max_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma) + \min_{\sigma\in L} \sum_{\omega\in G} P_M^k(\sigma,\omega)]$$

These bounds are used to provide the necessary and sufficient convergence conditions for a GHC algorithm to converge in probability to G, as given by Theorem 1.

Theorem 1 (Sullivan and Jacobson 2001): For a GHC algorithm at macro iteration k (fixed), with macro iteration transition matrix P_M^k,

a) (Necessary Condition)
If $\Pi_\omega^k \to 0$ as $k \to +\infty$, for all $\omega \in L$, then
$[\min_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma)] / [\max_{\sigma\in L} \sum_{\omega\in G} P_M^k(\sigma,\omega)] \to 0$ as $k \to +\infty$,

b) (Sufficient Condition)
If $[\max_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma)] / [\max_{\omega\in G} \sum_{\sigma\in L} P_M^k(\omega,\sigma) + \min_{\sigma\in L} \sum_{\omega\in G} P_M^k(\sigma,\omega)] \to 0$
as $k \to +\infty$, then $\Pi_\omega^k \to 0$ as $k \to +\infty$, for all $\omega \in L$.

Theorem 1 uses the macro iteration structure and the macro iteration transition matrix described above to provide conditions that can be used to determine when a GHC algorithm does or does not converge. These conditions are being studied and analyzed to determine convergence rates for different classes of GHC algorithms, as well as to determine how different GHC algorithm formulations can be compared and evaluated for various discrete optimization problems.

3. Ordinal Hill Climbing Algorithms

Ordinal hill climbing algorithms were introduced and analyzed during this grant. This section describes results in formulating and applying ordinal hill climbing algorithms to manufacturing process design optimization problems (Fischer et al. 1997) of interest to the Air Force. This effort has been interdisciplinary, involving researchers from the Materials Process Design Branch of the Air Force Research Laboratory (Wright Patterson Air Force Base), and Austral Engineering and Software, Inc., (Athens, Ohio). Note that the research and algorithm development work by our research group has been transitioned to Austral Engineering and Software, Inc., hence supporting their effort under a Phase II SBIR through the Air Force. This collaboration between an Air Force Laboratory, and industrial partner, and an academic institution has been highly productive and extremely synergistic in identifying problems of interest to the Air Force, as well as formulating and developing solutions to address such problems.

An important applied development has been the formulation of a hybrid algorithm that crosses ordinal optimization with generalized hill climbing algorithms. The resulting new algorithm is termed ordinal hill climbing. The results with this algorithm are reported in Sullivan and Jacobson (2000). To describe the ordinal hill climbing algorithm, ordinal optimization must be described.

Ordinal optimization (Ho et al. 1992) is a search procedure that can be used to obtain optimal/near-optimal designs for discrete manufacturing process design optimization problems. The strength of the ordinal optimization approach is in its focus on finding good designs, rather than trying to find the very best design (i.e., goal softening) (Lee et al. 1998). This allows ordinal optimization to reduce sampling over a very large design space (when searching for an optimal design) to sampling over a smaller, more manageable, set of good designs (see Chen and Kumar 1996 for an application of ordinal optimization to a discrete optimization problem). The strength of the generalized hill climbing algorithm is in its intelligent exploration of the design space when searching for an optimal design. In practice, generalized hill climbing algorithms find good designs in finite time, similar to ordinal optimization.

Ordinal Optimization is an algorithmic approach for identifying good designs over a large design space. By relaxing the requirement to obtain the optimal design, hence searching for designs that are in a small percentage of the best designs, the algorithm can efficiently weed out very poor designs and identify a set of designs (i.e., the selected subset) among which there exists a good design (i.e., an element of the good enough subset). The quality of the selected subset with respect to the good enough subset is obtained using an *alignment probability*, which measures the probability that these two subsets contain some integer value number of overlapping (common) designs. The very nature of relaxing the optimization requirement in this way has suggested the term *soft optimization* as appropriate for describing the execution of ordinal optimization.

By relaxing (or softening) the optimization requirement for ordinal optimization, a fast convergence rate for the algorithm can be obtained. In particular, Dai (1996) proves that the ordinal optimization convergence rate is exponential in the number of designs selected. He shows this by considering a set of t designs, selected independently and identically over the entire design space, and defining the *correct selection* (CS) event as the event in which the best design (over the t selected) is indeed the best overall design. Then, under mild restrictions, P{CS} is shown to be bounded below by $1-\alpha e^{-\beta t}$, where α and β are positive constants. Convergence can also established by showing that the alignment probability converges to zero. These results show that relaxing an algorithm's requirement from finding the overall best design, to finding a design in some top percentile of all designs in the design space, results in a significantly improved rate of convergence.

Using the strengths of both ordinal optimization and generalized hill climbing algorithms, a new algorithm is proposed, termed *ordinal hill climbing*. The ordinal hill climbing algorithm is a particular type of generalized hill climbing algorithm, with the hill climbing component of the generalized hill climbing algorithm based on ordinal information, where a set of designs are collected and evaluated at each iteration (similar to how ordinal optimization is applied). In particular, the algorithm is initialized with a set (of M) selected designs, chosen among all the designs in the design space (e.g., randomly). At each iteration k, a hill climbing random variable, R_k, defined over the discrete values {1, 2, ...,M} is generated ($R_k = r_k$) and used to keep the best r_k designs in the selected set, where the designs in the selected set are ordered from smallest to largest cost, with best referring to the designs with smallest cost. The $M-r_k$ open spots in the selected set are then filled with new designs, chosen among all the designs in the design space, or using some rule based on the designs that were kept in the selected set. This process is iteratively repeated until the algorithm terminates (e.g., a fixed number of iterations are executed). The ordinal hill climbing algorithm is described in pseudo-code form, given a design space, Ω, and a cost function (f) and a neighborhood function η (if needed) defined over the design space.

```
Select a set of M initial designs $D(0) \subset \Omega$
Set the iteration number k = 0
Define the hill climbing variable $R_k$, where
        i) $\Sigma^{M}_{m=1} P\{R_k = m\} = 1$,        ii) $R_0 = 1$ with probability one,
Repeat
        Order the designs in $D(k)$ from smallest to largest cost function values
        Generate $R_k (= r_k)$
        Keep the best (smallest cost function value) $r_k$ designs from the set $D(k)$. Call this set $E_1$.
        Generate M-$R_k$ new designs from the design space $\Omega$. Call this set $E_2$.
        Set $D(k+1) \leftarrow E_1 \cup E_2$.
        $k \leftarrow k+1$
Until stopping criterion is met
```

There are several parameters that must be initialized for the ordinal hill climbing algorithm. First, the value of M, the size of the selected design sets, *D(k),* must be set. Second, the initial set of M manufacturing process designs, $D(0)$, can be selected randomly or using any specified selection rule. In addition, there are two features of the algorithm where the user has great flexibility. First, the hill climbing variable R_k can be defined as any discrete (random or deterministic) non-negative variable defined over the set of integers (from one to M). Second, at each iteration, the method of filling in the $M- r_k$ designs in the selected set must be specified (e.g., generate neighbors of each of the designs that are kept in the selected set).

The intuition behind ordinal hill climbing algorithms is that at each iteration, the best designs, where best is measured using the value of the cost function (smaller cost designs are better than larger cost designs), are more likely to become part of, and remain in, the selected design set *D*(k). The hill climbing variable determines the acceptance or rejection of designs in the design set based on their relative costs (i.e., ordinal rank). Therefore, the *ranking* of the design costs, rather than the costs themselves, determines whether a design is accepted. For example, if R_k is close to one with high probability, as in the early stages of the algorithm execution, then most of the designs will be rejected from the selected set during this phase, hence guaranteeing a broad (and aggressive) sampling of designs. On the other hand, if as the algorithm progresses R_k is more likely to be close to M, then the algorithm will tend to retain good designs that have managed to remain in the selected design set.

In light of this observation, a possible third requirement on the hill climbing variable is $R_k \rightarrow M$ (with probability one) as $k \rightarrow +\infty$. As the algorithm proceeds, this requirement ensures that the algorithm will accept an increasing number of designs, until at termination, when $P\{\lim_{k\rightarrow+\infty} R_k = M\} = 1$, the algorithm accepts all of the designs in the selected design set. The defining feature of the ordinal hill climbing algorithm is that at termination, the final set of designs is likely to contain a good design. To identify a subset of this final set of designs to use, ordinal optimization can be applied.

There are a number of ways to simplify and/or fine-tune the ordinal hill climbing algorithm. If M is set to one at all iterations, then each iteration considers only a single design. This simplification reduces the ordinal hill climbing algorithm to a generalized hill climbing algorithm. There are an unlimited number of choices for the hill climbing variable, R_k, that satisfy the requirements listed in the ordinal hill climbing pseudo-code. There are also numerous ways in which the selected design set can be updated from *D*(k) to *D*(k+1), depending, for example, on the choice of neighborhood function. To illustrate, this update can be done using multiple neighbors of only the best, or near-best (e.g., top $\beta * M$, where $\beta > 0$ close to zero) designs. Stopping criterion that can be used for the ordinal hill climbing algorithm include looking at the top αM designs, with $\alpha > 0$ close to zero, and stopping the algorithm if this top αM set of designs does not change over some specified number of iterations. Each of these modifications results in a large selection of ordinal hill climbing algorithm variations that can be used to address the an integrated blade rotor discrete manufacturing process design optimization problem of interest to the Air Force. Computational results with the ordinal hill climbing algorithm applied to this problem are reported in Sullivan and Jacobson (2000).

An important development during this phase of the project was the identification of a relationship between genetic algorithms and ordinal hill climbing algorithms. Genetic algorithms (GA) are an optimization strategy that has been successfully applied to numerous discrete optimization problems. The foundation of GA is derived from the Darwinian notion of "survival of the fittest" (Reeves 1993); this notion suggests that as time evolves, parents produce new offspring that are more acceptable than members of previous populations. In other words, over a period of time, parents are continually mating to produce successive generations of children that are closer to optimality than their predecessors.

The GA ordinal component of comparing cost function values over a set of children (i.e., designs) suggest that there may be a natural bridge between GA and ordinal hill climbing algorithms. Though GA have proven to be effective for addressing intractable discrete optimization problems, and can be classified as a type of hill climbing approach, its link with generalized hill climbing algorithms (through the ordinal hill climbing formulation) provides a well-defined relationship with other generalized hill climbing algorithms (like simulated annealing and threshold accepting). Therefore, such an analysis provides useful insights and observations that may fuel further research into both ordinal hill climbing and genetic algorithms, and how they fit together on a broader scale.

The bridge between GA and the ordinal hill climbing algorithm framework is defined through the hill climbing variable R_k and the method by which each successive selected subset is updated (based on a neighborhood function definitions). In particular, the hill climbing variable R_k for ordinal hill climbing algorithms determines the number of parents carried over from the selected set (population) $D(k)$ to the set E_1. Once the set E_1 has been determined, its members can be mated (e.g., according to a crossover rule) to produce $M-R_k$ offspring. Therefore, the GA concept of mating serves as the neighborhood function in the ordinal hill climbing algorithm framework. This set of $M-R_k$ offspring, E_2, together with the set E_1 becomes the next selected set or population (i.e., $D(k+1) = E_1 \cup E_2$).

Basic GA consist of three components:

i) Reproduction
ii) Crossover
iii) Mutation

Reproduction is the process by which individual parents are evaluated for mating and inclusion in future populations. Variants of GA can be described by employing different hill climbing variables R_k (i.e., the selection of parents to be contained in E_1 varies with the choice of R_k). Each parent in a population is evaluated according to its cost function value. In general, parents with good (lower) cost function values are more likely to be among the R_k selected parents for the set E_1. However, for diversification, the set E_1 can be a mix of both good and bad parents. *Crossover* is a process by which offspring are generated based upon the cost function values of the parents. A particular crossover method needs to be defined before the GA is executed. Crossover may consist of generating a neighbor of a parent in E_1 to produce offspring or combining different parts of two parents in E_1 to produce offspring.

The following is the ordinal hill climbing algorithm formulation of *simple* GA, which uses reproduction and crossover components:

```
Select a set of M initial designs $D(0) \subset \Omega$
Set the iteration number $k = 0$
Repeat
    Order the designs in $D(k)$ from smallest to largest cost function values
    Generate $R_k$ $(= r_k)$
    Keep the best (smallest cost function value) $r_k$ designs from the set $D(k)$ as parents for creating offspring.
        Call this set $E_1$.
    Generate offspring of $E_1$ according to the crossover rule to obtain $M-r_k$ new designs. Call this set $E_2$.
    Set $D(k+1) \leftarrow E_1 \cup E_2$.
    $k \leftarrow k+1$
Until stopping criterion is met
```

Mutation involves the random alteration of a parameter value in the offspring. Mutation can play either a primary or secondary role in GA, depending on the desired aggressiveness of GA. If the entire population has cost function values that are relatively close to each other, the search may easily get caught at a local optimum. In this situation, it may be desirable to increase the probability of mutation to ensure the inclusion of designs in the set E_1 sufficiently different from those already in the population.

The following is the ordinal hill climbing formulated simple GA with a mutation component.

```
Select a set of M initial designs $D(0) \subset \Omega$
Set the mutation probability
Set the iteration number $k = 0$
Repeat
    Order the designs in $D(k)$ from smallest to largest cost function values
    Generate $R_k$ $(= r_k)$
    Keep the best (smallest cost function value) $r_k$ designs from the set $D(k)$ as parents for creating
        offspring. If the largest and smallest cost function values of parents among these $r_k$ values are
        relatively close to each other, increase the mutation probability, hence the diversity of the parents
        set. Call this set $E_1$.
    Generate offspring of $E_1$ according to the crossover rule to obtain $M-r_k$ new designs. Call this set $E_2$.
    Set $D(k+1) \leftarrow E_1 \cup E_2$.
    $k \leftarrow k+1$
Until stopping criterion is met
```

At the end of each iteration, the new population becomes the population of parents that will be used to perform reproduction and crossover to generate yet another new population. This process is repeated until the algorithm terminates; at termination, the final population will (hopefully) contain an optimal parent (i.e., an optimal design).

These ordinal hill climbing algorithm formulations for GA illustrate the power and flexibility of the ordinal hill climbing algorithm framework. It should be noted that by defining the procedure by which the selected subset in ordinal hill climbing algorithms is updated without a neighborhood function, it is possible to obtain an ordinal hill climbing algorithm formulation that is not a GA (such as by simply randomly generating designs to obtain set E_2). Moreover, problem-specific implementations for GA may not fit into the ordinal hill climbing framework. Further research is needed to fully assess the relationship between ordinal hill climbing algorithms and GA. However, the two formulations presented provide a first step towards developing a bridge between GA and other search strategies like simulated annealing, threshold accepting, and tabu search (Johnson 1996, Jacobson et al. 1998) using the generalized hill climbing paradigm (Johnson and Jacobson 2001a,b). Moreover, these formulations serve to illustrate the power of the ordinal optimization strategy in addressing deterministic discrete optimization problems.

4. Multiple Sequence Optimization Using Generalized Hill Climbing Algorithms

Multiple sequence optimization using generalized hill climbing algorithms were introduced to simultaneously address sets of manufacturing process design optimization problems (Vaughan et al. 2000). Initial results with generalized hill climbing algorithms required the manufacturing process design sequence to be fixed, with the generalized hill climbing algorithm used to identify optimal input parameter settings. This section describes a new neighborhood function that allows generalized hill climbing algorithms to be used to also identify the optimal discrete manufacturing process design sequence among a set of valid design sequences. The neighborhood function uses a switch function for all the input parameters, hence allows the generalized hill climbing algorithm to simultaneously optimize over both the design sequences and the inputs parameters. Computational results are reported with an integrated blade rotor discrete manufacturing process design problem under study at the Materials Process Design Branch of the Air Force Research Laboratory, Wright Patterson Air Force Base (Dayton, Ohio, USA). This research provided motivation for the development of the simultaneous generalized hill climbing algorithm framework (see Vaughan and Jacobson 2001) discussed in Section 5.

To describe the discrete manufacturing process design optimization problem, and how multiple sequence optimization using generalized hill climbing algorithms can be used to address the problem, Jacobson et al. (1998) present an extensive problem description. For completeness, this information is also presented here. First, several definitions are needed.

Let the manufacturing processes be denoted by $P_1, P_2, \ldots, P_n$. Associated with each process are (continuous or discrete) *controllable input parameters*, *uncontrollable input parameters*, and *output parameters*. The output parameters for a particular process may serve as the uncontrollable input parameters for a subsequent process. A sequence of processes, together with a particular set of input parameters constitutes a *manufacturing process design*; label such designs $D_1, D_2, \ldots, D_N$. Note that if one (or more) controllable input parameter is continuous, then $N = +\infty$. Otherwise, $N < +\infty$. Without loss of generality, assume that all the controllable input parameters are discrete, since any continuous controllable input parameter can be discretized over an arbitrarily fine grid. Under this assumption, the number of manufacturing process designs is finite, though potentially very large.

Denote the *design space* by Ω, the set of all manufacturing process designs (i.e., $\Omega = \{D_1, D_2, \ldots, D_N\}$), where a subset of the designs in Ω are feasible. Infeasible designs violate prespecified constraints on the manufacturing processes and the unit being manufactured, including geometric and microstructural properties, and constraint violations on the output parameters (e.g., the required forging press pressure may not exceed its upper bound limitations). Define a *cost function* $f: \Omega \rightarrow [0,+\infty)$ that assigns a non-negative value to each element of the design space, where cost includes monetary costs and measures for how well the finished unit meets prespecified geometric and microstructural properties. Penalties for constraint violations on the output parameters and measures that ensure a robust manufacturing design (i.e., the manufacturing process design is stable) are included in the cost function. Define a *neighborhood function* $\eta: \Omega \rightarrow 2^{\Omega}$, where $\eta(D) \subset \Omega$ for all $D \in \Omega$. The neighborhood function establishes connections between the designs in the design space (either through the controllable input parameters or through the design sequences), hence allows the design space to be traversed or searched by moving between designs. For all solutions in the solution space, an individual neighbor can be generated using generation probabilities (i.e., a probability mass function) among all possible neighbors, as defined by the neighborhood function η (see Johnson and Jacobson 1999 for complete details). The goal is to identify the globally optimal manufacturing process design D^* (i.e., $f(D^*) \leq f(D)$ for all $D \in \Omega$), or more realistically, near-optimal designs.

A manufacturing process design is needed to transform a billet into an integrated blade rotor (IBR) geometric shape (Gunasekera et al. 1996). Five manufacturing process design sequences have been identified that can achieve this transformation. Define the following notation for the seven processes that make up these five designs:

P_0 is the cast ingot process
P_1 is the extrusion process
P_2 is the upset process
P_3 is the machine preform process
P_4 is the blocker forge process
P_5 is the rough machining process
P_6 is the finished shape process

The five possible manufacturing process design sequences, provided by researchers at the Materials Process Design Branch of the Air Force Research Laboratory and Ohio University, are

$$P_0P_2P_4P_6 \qquad P_0P_2P_5P_6 \qquad P_0P_2P_3P_4P_6 \qquad P_0P_1P_2P_4P_6 \qquad P_0P_1P_3P_4P_6 .$$

Associated with each of the seven processes are uncontrollable and controllable input parameters, and output parameters. For example, for process P_0, there are two controllable input parameters (the radius of the billet and the height of the billet), zero uncontrollable input parameters, and two output parameters, which are just the controllable input parameter values. See the Appendix for a list of the controllable input parameters for each of the seven processes. Fischer et al. (1997) and Gunasekera et al. (1996) present complete details on all seven processes and their parameters.

Associated with each controllable input parameter is a discrete (naturally, or discretized from a continuous domain) set of feasible values (the Appendix contains a listing of these values for all the controllable input parameters for the seven processes). The cost function quantifies the cost associated with not meeting certain geometric and microstructural properties of the finished product, the monetary cost in producing the finished product, and cost penalties for constraint violations (see Jacobson et al. 1998). The goal is to identify a valid manufacturing process design sequence, together with values for all the controllable input parameters for the processes, that result in a feasible manufacturing process design that produces the IBR unit at total minimum cost.

Computer simulation models of the manufacturing processes described above have been developed (Fischer et al. 1997, Gunasekera et al. 1996). This moves the search for an optimal manufacturing process design from the shop floor (where trial and error has typically been applied, using actual materials) to a computer platform. However, even using high speed computing resources, the search for an optimal manufacturing process design may take a prohibitive amount of time. To circumvent this problem, generalized hill climbing algorithms have been introduced as a tool to be used with the computer simulation models to identify optimal/near-optimal manufacturing process designs.

Jacobson et al. (1998) uses three different neighborhood functions for GHC algorithms to identify optimal controllable input parameter values for a set of fixed valid manufacturing process design sequences. By considering each design sequence individually, they solve for the controllable input parameter values that minimize the cost, and then choose the overall minimum cost design sequence (and associate optimal controllable input parameter values) as the optimal design. For the five design sequence problems described previously, this is a reasonable approach. However, for complex parts, where the number of design sequences may be very large, performing such optimization (one design sequence at a time) does not lend itself to efficient, automated optimization procedures. Moreover, such an approach does not exploit any common manufacturing process subsequence components within two or more design sequences, which can lead to added efficiencies in identifying optimal controllable input parameter values.

To overcome these problems, applying GHC algorithms to optimize *between* (or across) design sequences requires a new neighborhood function that captures the cost differences among the different process design sequences, but also allows for seamless transitions (using a well-defined mechanism) between such sequences. To formulate such a neighborhood function requires several new definitions and notations.

Define S to be the set of possible manufacturing process design sequences and W to be the set of possible values for the controllable input parameters for the seven processes. Therefore, each design D can be represented as a two-tuple (**w**, **s**) where $\mathbf{w} \in W$ and $\mathbf{s} \in S$. The neighborhood function is denoted by $\eta(D) = \eta(\mathbf{w}, \mathbf{s}) = (\eta_1(\mathbf{w}), \eta_2(\mathbf{s}))$, where $\eta_1: W \rightarrow 2^W$ and $\eta_2: S \rightarrow 2^S$. The neighborhood function η allows the GHC algorithm to simultaneously change both the input parameters and the manufacturing process design sequence.

There are numerous ways to define the neighborhood functions η_1 and η_2. For the experiments reported in the computational section, for each $\mathbf{w} \in W$

$\eta_1(\mathbf{w}) = \{\mathbf{w}' \in W \mid \mathbf{w}'$ and $\mathbf{w}$ have *at most* one controllable input parameter different for each process$\}$.

To define the neighborhood function η_2 on the set S, the process design sequences can be represented using *binary activity* vectors. Given a process design sequence $\mathbf{s} \in S$, let

$$V_i = \begin{cases} 1, \text{if } P_i \text{ is contained in s} \\ 0, \text{otherwise} \end{cases}.$$

The process design sequence s can be represented by the binary activity vector $\mathbf{s} \in R^5$ where

$$\mathbf{s} = (V_1, V_2, V_3, V_4, V_5).$$

Note that every manufacturing process design sequence begins with P_0 (cast ingot) and ends with P_6 (finish machining), hence each such sequence can be represented by a binary activity vector of length five. To define such vectors, a strict precedence relation must be imposed on the order in which the processes can occur. For the seven processes, the precedence relation is given as $P_0 <. \ P_1 <. P_2 <. P_3 <. P_4 <. P_5 <. P_6$, where $P' <. P''$ means process P' must occur before process P" when both P' and P" occur in the same manufacturing process design sequence. Therefore, the set of *possible manufacturing process* design sequences can be formally defined using the binary activity vectors,

$$S = \{\mathbf{s} \in R^5 \mid \mathbf{s} \text{ is a binary activity vector of length five}\}.$$

For the computational results section, S contains thirty-two possible manufacturing process design sequences. Of these, only the five manufacturing process design sequences depicted in the backgrounds section are considered valid. These five manufacturing process design sequences, denoted as $S_v \subseteq S$, are referred to as *valid manufacturing process design sequences.* Table 1 depicts these five valid manufacturing process design sequences and their corresponding binary activity vectors.

TABLE 1
Binary Activity Vectors for Valid Manufacturing Process Design Sequences

Process Sequence	Binary Activity Vector
$P_0P_2P_4P_6$	(0,1,0,1,0)
$P_0P_2P_5P_6$	(0,1,0,0,1)
$P_0P_2P_3P_4P_6$	(0,1,1,1,0)
$P_0P_1P_2P_4P_6$	(1,1,0,1,0)
$P_0P_1P_3P_4P_6$	(1,0,1,1,0)

Using the binary activity vector representation for elements of S, the neighborhood function η_2 can be implemented in a GHC algorithm with a random variable that maps elements of S to elements of S. The neighborhood function η_2 depends on a *probability switch* vector. The components of a probability switch vector $\mathbf{q} = (q_1, q_2, q_3, q_4, q_5)$ are the probabilities that a particular process is switched from active (one) to inactive (zero) or from inactive (zero) to active (one). In order to generate a neighbor from $\eta_2(\mathbf{s})$, $\mathbf{s} \in S_v$, the components of the probability switch vector $\mathbf{q}$ are permuted to the vector $\mathbf{q'}$, with the neighbor generated by switching each component of $\mathbf{s}$, namely s_i, i = 1,2,3,4,5, with probability q'_i. Note that if one or more of the q_i, i = 1,2,3,4,5, are zero, the neighborhood function may not contain the entire set of thirty-two possible design sequences. Therefore, for each $\mathbf{s} \in S_v$, setting one or more of the q_i, i = 1,2,3,4,5, to zero can reduce the number of possible manufacturing process design sequences in $\eta_2(\mathbf{s})$.

To switch a process from active to inactive or vice versa, define a set of random variables $\delta_p:\{0,1\} \rightarrow \{0,1\}$ where $p \in [0,1]$ and

$$\delta_p(x) = \begin{cases} x & \text{with probability } 1-p \\ 1-x & \text{with probability } p \end{cases}. \qquad (1)$$

Define the *permutation random variable*, Π_j: $[0,1]^5 \rightarrow [0,1]^5$, which permutes the components of a vector in $[0,1]^5$. The permutation random variable will be used to permute the components of a probability switch vector. The permutation random variable is defined so that every permutation $\mathbf{p'}$ of $\mathbf{p} = (p_1, p_2, p_3, p_4, p_5)$ occurs with equal probability. Therefore, at iteration j,

$$P\{\Pi_j(\mathbf{p}) = \mathbf{p'}\} = 1 / 5!$$

for all permutations $\mathbf{p'}$ of $\mathbf{p}$, hence a potential neighboring solution of $\mathbf{s}$ can be obtained using (1) as

$$\mathbf{s'} = (\delta_{p'_1}(s_1), \delta_{p'_2}(s_2), \delta_{p'_3}(s_3), \delta_{p'_4}(s_4), \delta_{p'_5}(s_5)),$$

where $\Pi_j(\mathbf{q}) = \mathbf{q'} = (q'_1, q'_2, q'_3, q'_4, q'_5)$ and $\mathbf{q}$ is the probability switch vector. Note that if $\mathbf{s'} \notin S_v$, then the neighboring solution of **s** is generated to be **s**.

Given the probability switch vector **q**, let h be the number of non-zero elements of **q**. Then neighborhood function, η_2, is defined as

$$\eta_2(\mathbf{s}) = \{\mathbf{s'} \in S \mid \mathbf{s'} \text{ has at most } h \text{ binary components different from } \mathbf{s}\}.$$

Note that for this neighborhood function, the generation probabilities are typically not uniform, since they depend on the probability switch vector.

The choice of probability switch vector **q** depends on $|S|$ (the cardinality of the possible process design sequences) compared to $|S_v|$ (the cardinality of the valid process design sequences). For the problem described in the Background section, the number of possible design sequences is $2^5 = 32$, which is large compared to the number of valid process design sequences. There is a trade-off between quickly moving between different manufacturing process design sequences and allowing the GHC algorithm to spend a large number of iterations exploring the various input parameter values in one particular design sequence. A neighborhood function that generates a limited number of design sequence changes may be too restrictive (myopic), hence prevent the GHC algorithm execution from visiting the (globally) optimal manufacturing process design sequence. This is the same problem that arises with many local search algorithms, namely how to choose the neighborhood function (i.e., should it be myopic, with small neighborhoods, or aggressive, with large neighborhoods.)

To apply GHC algorithms to multiple design sequences, a mathematical structure to compute the probability of moving between the five valid manufacturing process design sequences, using the neighborhood function, η_2, must be developed. To define this structure, let $\Psi: [0, 1]^5 \times S \rightarrow S$ determine a neighbor of **s** (i.e., $\mathbf{s'}$) by first permuting **p** (i.e., by applying $\Pi_j(\mathbf{p})$) and then generating $\mathbf{s'} = (\delta_{p'_1}(s_1), \delta_{p'_2}(s_2), \delta_{p'_3}(s_3), \delta_{p'_4}(s_4), \delta_{p'_5}(s_5))$. Note that references to an iteration implies an application of $\Psi(\mathbf{p}, \mathbf{s}) = s'$. Moreover, recall that the possible design sequences are the $32 = |\{0, 1\}|^5 = 2^5$ elements in the range of $\Psi(\mathbf{p}, \mathbf{s})$, and that of these 32 possible design sequences, only 5 represent valid manufacturing process design sequences.

To formally define the distance between manufacturing process design sequences, consider the metric space $<\Sigma^n, \rho>$, with $\Sigma = \{0,1\}$, where the metric ρ is defined on $\Sigma^n \times \Sigma^n$ such that the *distance* between two possible manufacturing process design sequences $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \Sigma^n$ and $\mathbf{t} = (t_1, t_2, \ldots, t_n) \in \Sigma^n$ is

$$\rho(\mathbf{s}, \mathbf{t}) = |s_1 - t_1| + \ldots + |s_n - t_n| \tag{2}$$

(Royden, 1988, p.140). To illustrate this metric, the distance between $P_0P_1P_2P_4P_6$ (with activity vector **s** = (1, 1, 0, 1, 0)) and $P_0P_1P_3P_4P_6$ (with activity vector **t** = (1, 0, 1, 1, 0)) is $\rho(\mathbf{s}, \mathbf{t}) = 2$. Define

$$E = \{\mathbf{s} \in \Sigma^n \mid \rho(\mathbf{s}, \mathbf{0}) = 1\},$$

where $\varepsilon^h \in E$ such that $\varepsilon^h = (0, \ldots, 0, 1, 0, \ldots, 0)$ with the one appearing in the hth position.

Define a *binary switch* on the jth element of an activity vector **s** to be the modulus two addition of $\varepsilon^j \in E$ with **s**. For example, consider the activity vector **s** = (1, 1, 0, 1, 0) corresponding to $P_0P_1P_2P_4P_6$. By performing a binary switch on the second element of **s** (i.e., by adding $\varepsilon^2 = (0, 1, 0, 0, 0)$), the resulting activity vector is $\mathbf{s}' = \mathbf{s} + \varepsilon^2 = (1, 0, 0, 1, 0)$, corresponding to $P_0P_1P_4P_6$. By performing a second binary switch (i.e., by adding $\varepsilon^3 = (0, 0, 1, 0, 0)$ to $\mathbf{s}'$), the resulting activity vector is $\mathbf{t} = \mathbf{s}' + \varepsilon^3 = (1, 0, 1, 1, 0)$, corresponding to $P_0P_1P_3P_4P_6$. Therefore, the metric $\rho(\mathbf{s}, \mathbf{t})$ represents the minimum number of binary switches required to move from **s** to **t**. For example, the distance between $P_0P_1P_2P_4P_6$ and $P_0P_1P_3P_4P_6$, represented as activity vectors **s** = (1, 1, 0, 1, 0) and **t** = (1, 0, 1, 1, 0), respectively, is $\rho(\mathbf{s}, \mathbf{t}) = 2$. This distance can also be obtained by observing that the minimal number of binary switches required to move from **s** to **t**. The distances between all five valid manufacturing process design sequences are given in Table 2.

TABLE 2
Distances Between Valid Manufacturing Process Design Sequences

	$P_0P_2P_4P_6$ (0,1,0,1,0)	$P_0P_2P_5P_6$ (0,1,0,0,1)	$P_0P_2P_3P_4P_6$ (0,1,1,1,0)	$P_0P_1P_2P_4P_6$ (1,1,0,1,0)	$P_0P_1P_3P_4P_6$ (1,0,1,1,0)
$P_0P_2P_4P_6$ (0,1,0,1,0)	0	2	1	1	3
$P_0P_2P_5P_6$ (0,1,0,0,1)	2	0	3	3	5
$P_0P_2P_3P_4P_6$ (0,1,1,1,0)	1	3	0	2	2
$P_0P_1P_2P_4P_6$ (1,1,0,1,0)	1	3	2	0	2
$P_0P_1P_3P_4P_6$ (1,0,1,1,0)	3	5	2	2	0

The domain of the neighborhood function η_2 is $S = \{\mathbf{s} \in \Sigma^n \mid \mathbf{s}$ is an activity vector of a valid manufacturing process design sequence$\}$, where $\eta_2(\mathbf{s}')$ (for a fixed $\mathbf{s}' \in S$) is denoted by $\Sigma_i \subseteq \Sigma^n$. For all $\mathbf{s} \in \Sigma_i$, the probability that $\Psi(\mathbf{p}, \mathbf{s}') = \mathbf{s}$ given that $\rho(\mathbf{s}', \mathbf{s}) = k$ (see (2)) is

$$P\{\Psi(\mathbf{p}, \mathbf{s}') = \mathbf{s}\} = P\{\rho(\mathbf{s}', \mathbf{s}) = k\} \Big/ \binom{5}{k} \text{ for } \mathbf{s} \text{ such that } \rho(\mathbf{s}', \mathbf{s}) = k.$$

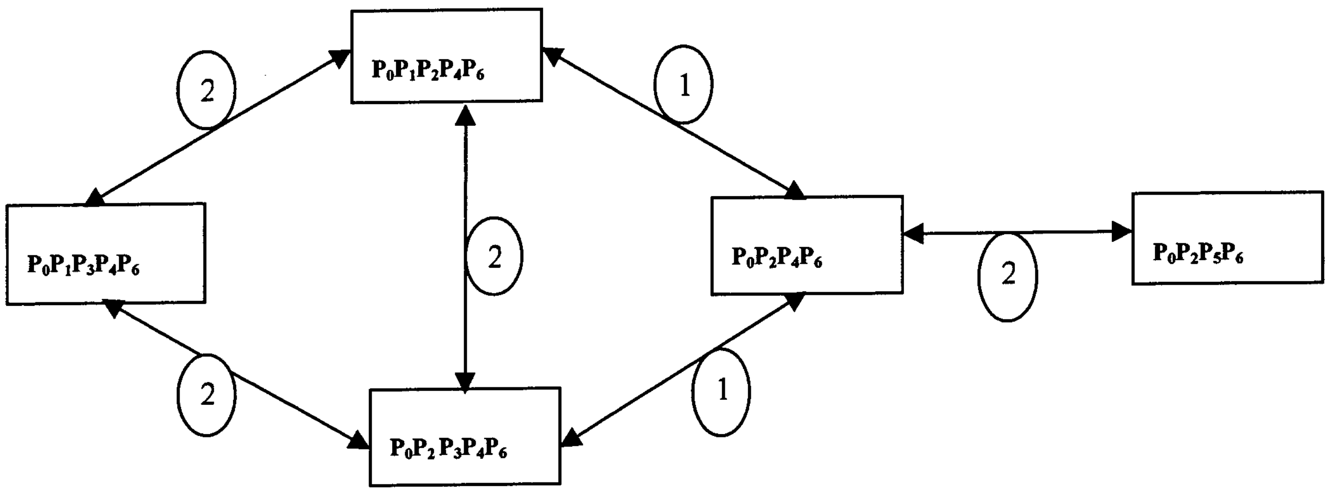
To determine P{ρ(**s′**, **s**) = k}, recall that Π(**p**) = **s′** is a sequence of binary switches on **s**, where the cardinality of binary switches (the previously defined distance) is dependent on **p** = (p_1, p_2, p_3, p_4, p_5). Table 3 provides the probabilities that ρ(**s**, **s′**) = k for k = 1, 2, 3, 4, 5.

To illustrate the computation of these probabilities, if η_2 is defined with **p** = (.5, 0, 0, 0, 0), then Π(**p**) results in no binary switches with probability .5, and one binary switch with probability .5. If η_2 is defined with **p** = (.5, .5, 0, 0, 0), then Π(**p**) results in no binary switches with probability .25, one binary switch with probability .5, and two binary switches with probability .25. The five valid manufacturing process design sequences can be traversed using **p** = (p_1, p_2, 0, 0, 0). For neighborhood function η_2, with **p** = (p_1, p_2, 0, 0, 0), it is possible to move between valid design sequences (in a single iteration) with activity vectors that are at most two binary switches apart. Figure 1 depicts movement between sequences in a single iteration, where the nodes denote the five valid manufacturing process design sequences, and the edge values denote the distance between the sequences at the corresponding nodes.

TABLE 3
Distance Probabilities

k	P{ρ(**s**, **s′**) = k}
1	$\sum_{j=1}^{5} p_j \prod_{\substack{i=1\\ i\neq j}}^{5} (1-p_i)$
2	$\sum_{i=1}^{5} p_j \sum_{\substack{j=1\\ j\neq i}}^{5} p_j \prod_{\substack{k=1\\ k\neq j\\ k\neq i}}^{5} \frac{(1-p_k)}{2}$
3	$\sum_{i=1}^{5} (1-p_j) \sum_{\substack{j=1\\ j\neq i}}^{5} (1-p_j) \prod_{\substack{k=1\\ k\neq j\\ k\neq i}}^{5} \frac{p_k}{2}$
4	$\sum_{j=1}^{5} (1-p_j) \prod_{\substack{i=1\\ i\neq j}}^{5} p_i$
5	$p_1 p_2 p_3 p_4 p_5$

FIGURE 1: The Five Valid Manufacturing Process Design Sequences



For the probability switch vector **p** = (p_1, p_2, 0, 0, 0), where $0 < p_1, p_2 < 1$, the probability of moving between sequences at any given iteration is a function of the number of binary switches required to move between these sequences. In particular, define the two *travel probabilities*

$$P\{1\text{-binary switch}\} = \left(\frac{p_1(1-p_2)}{\binom{5}{1}} + \frac{p_2(1-p_1)}{\binom{5}{1}}\right) = \frac{(p_1(1-p_2)+p_2(1-p_1))}{5}$$

and

$$P\{2\text{-binary switches}\} = \frac{p_1p_2}{\binom{5}{2}} = \frac{p_1p_2}{10}.$$

These expressions are validated empirically using Monte Carlo search, by recording the number of times the algorithm iterated from a source design sequence (say A) to a neighboring design sequence (say B) and then dividing by the number of times the algorithm visited the source design sequence (A). Table 4 provides the Monte Carlo search computational results for **p** = (p_1, p_2, 0, 0, 0, 0) = (.9, .9, 0, 0, 0), with 100,000 iterations. Note that for Monte Carlo search, all iterations are outer loop iterations with one inner loop iteration per outer loop iteration (i.e., K = 100,000 and M(k) = 1, k = 1,2,…,100,000). Ninety percent confidence intervals (CI) are reported for the travel probabilities, where the confidence intervals marked with an asterisk are those that cover the true travel probability values (of these twelve confidence intervals, nine cover the true travel probability value).

TABLE 4
Traveling Between Sequences, p = (.9, .9, 0, 0, 0)

Sequence A	Sequence B	Number of Binary Switches	True Travel Probability	Estimated Travel Probability	True Travel Probability CI
$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	2	.081	.079	(.0769, .0811)*
$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	2	.081	.083	(.0809, .0851)*
$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	2	.081	.076	(.0728, .0792)
$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	2	.081	.078	(.0748, .0812)*
$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	1	.036	.038	(.0358, .0402)*
$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	2	.081	.078	(.0748, .0812)*
$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	2	.081	.082	(.0788, .0852)*
$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	1	.036	.034	(.0318, .0362)*
$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	1	.036	.033	(.0309, .0351)
$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	$P_0P_2P_5P_6$ (0, 1, 0, 0, 1)	2	.081	.082	(.0789, .0851)*
$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	1	.036	.034	(.0319, .0361)*
$P_0P_2P_5P_6$ (0, 1, 0, 0, 1)	$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	2	.081	.089	(.0858, .0922)

Computational results are reported with the neighborhood function that allows for optimization between the manufacturing process design sequences. The results in Jacobson et al. (1998) suggest that the Weibull accepting hill climbing random variable was the most effective (among five different GHC algorithm formulations tested) for optimizing over the controllable input parameters for a particular design sequence, hence was used to define η_1. The objective in running these experiments is to assess the performance of the GHC algorithm formulations using the neighborhood functions (η_1, η_2), as well as to identify optimal/near optimal manufacturing process design sequences and input parameters using computer simulation models.

For Weibull accepting, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β_1, where $0 \le \beta_1 \le 1$ (i.e., $t_k = \beta_1 t_{k-1}$). The initial temperature parameter is $t_0 =$ 10,000, with $\beta_1 = .96$ and shape parameter $\alpha = 2.0$. The acceptance probability for the Weibull accepting GHC algorithm is

$$P\{R_k(D,D') \geq \delta\} = e^{\left(-\delta/t_k\right)^{\alpha}},\ D \in \Omega,\ D' \in \eta(D) \text{ for all } k = 1,2,\ldots,K. \tag{3}$$

Note that if $\alpha = 1$, then Weibull accepting reduces to simulated annealing (Jacobson et al. 1998).

The cost function evaluates the total cost associated with the simulated manufacturing process design, in US dollars. The initial cost is the cost of the initial billet, which depends on the dimensions of the billet and the specific metal being processed. The costs for the forging processes include:

i) set up costs,
ii) post-inspection costs,
iii) die wear costs,
iv) press run costs,
v) the cost of possible strain-induced-porosity damage in the workpiece.

Penalties are incurred with the forging processes when

i) the press capacity is exceeded,
ii) the aspect ratio of the workpiece is too large,
iii) the geometry of the workpiece conflicts with the die geometry.

The cost to machine the workpiece is the cost of the material removed from the workpiece, where a penalty cost is incurred when the geometry of the workpiece conflicts with the desired final geometry of the workpiece after machining. After the workpiece is processed, a mandatory ultrasonic non-destructive evaluation cost and, if necessary, a cost of heat treatment is accrued. In addition, the final microstructure of the workpiece is evaluated; if the microstructure violates predetermined specifications, a penalty cost is incurred. All penalties are translated into US dollars in the cost function.

Computational results with the Weibull accepting GHC algorithm incorporating the new neighborhood function (η_2) are reported. The Weibull accepting GHC algorithms were executed with different values of K and M = M(k), k = 1,2,...,K, as well as with varying values for the components of the probability switch vector **p**. Thirty (independently seeded) replications of each GHC formulation were made, each initialized with a different initial manufacturing process design sequence. The same thirty initial design sequences (for the thirty replications) were used across the different neighborhood functions (i.e., the different probability switch vectors). The initial controllable input parameter values for replications two through thirty were obtained by randomly selecting a neighbor (η_1) of the first replication's feasible controllable input parameters values. The mean (μ) and standard deviations (σ), as well as the minimum and maximum cost function values, were computed from the optimal values across these thirty replications. All computational experiments were executed on a SUN ULTRA-1 workstation (128 Mb RAM). Each set of thirty replications took approximately 30 CPU minutes.

In Tables 5-7, the variable γ represents the percentage of the replications that the GHC algorithm finds the optimal valid manufacturing process design sequence. The variable κ represents the average number of times, over the thirty replications, neighborhood function η_2 generates a manufacturing process design sequence that is different from the incumbent design sequence.

TABLE 5
GHC Algorithm Results, K = 50 and M = 200

Probability Switch Vector	γ	μ	σ	Minimum	Maximum	κ
(0.1, 0.1, 0, 0, 0)	9/30	2177.57	199.00	1935.71	2928.17	357.7
(0.2, 0.2, 0, 0, 0)	22/30	2047.62	209.89	1927.01	2928.72	538.6
(0.3, 0.3, 0, 0, 0)	20/30	2041.25	145.93	1919.28	2250.15	793.1
(0.4, 0.4, 0, 0, 0)	27/30	1973.88	92.00	1930.43	2245.28	1001.5
(0.5, 0.5, 0, 0, 0)	21/30	2033.21	140.17	1919.28	2248.08	1110.7
(0.6, 0.6, 0, 0, 0)	21/30	2033.88	142.08	1919.28	2254.63	1268.8
(0.7, 0.7, 0, 0, 0)	24/30	2009.45	118.78	1921.21	2250.15	1418.1
(0.75, 0.75,0, 0, 0)	25/30	1998.00	111.67	1927.01	2250.15	1408.2
(0.8, 0.8, 0, 0, 0)	25/30	1993.57	113.97	1921.21	2245.28	1438.4
(0.833, 0.833, 0, 0, 0)	27/30	1976.08	92.32	1927.01	2245.28	1402.4
(0.9, 0.9, 0, 0, 0)	27/30	1977.57	91.74	1921.21	2250.15	1443.1

TABLE 6
GHC Algorithm Results, K = 100 and M = 100

Probability Switch Vector	γ	μ	σ	Minimum	Maximum	κ
(0.1, 0.1, 0, 0, 0)	15/30	2089.41	156.13	1921.21	2245.28	389.3
(0.2, 0.2, 0, 0, 0)	15/30	2115.83	213.35	1921.21	2898.11	653.9
(0.3, 0.3, 0, 0, 0)	24/30	2000.99	125.00	1921.21	2248.08	800.9
(0.4, 0.4, 0, 0, 0)	25/30	1996.43	112.76	1919.28	2249.97	1133.6
(0.5, 0.5, 0, 0, 0)	25/30	1993.50	113.54	1921.21	2245.28	1285.8
(0.6, 0.6, 0, 0, 0)	22/30	2025.29	132.74	1923.70	2245.28	1389.0
(0.7, 0.7, 0, 0, 0)	24/30	2005.69	120.13	1932.86	2248.08	1498.0
(0.75, 0.75,0, 0, 0)	22/30	2024.25	135.71	1921.21	2250.15	1419.7
(0.8, 0.8, 0, 0, 0)	24/30	2004.95	121.77	1919.28	2244.25	1425.1
(0.833, 0.833, 0, 0, 0)	24/30	2008.17	120.15	1921.21	2245.28	1451.0
(0.9, 0.9, 0, 0, 0)*	27/30	1974.30	93.37	1921.21	2248.08	1397.9

TABLE 7
GHC Algorithm Results, K = 200 and M = 50

Probability Switch Vector	γ	μ	σ	Minimum	Maximum	κ
(0.1, 0.1, 0, 0, 0)	9/30	2196.01	249.69	1919.28	2959.29	344.7
(0.2, 0.2, 0, 0, 0)	11/30	2131.19	154.88	1923.15	2293.87	714.1
(0.3, 0.3, 0, 0, 0)	20/30	2037.88	149.32	1919.28	2257.53	799.7
(0.4, 0.4, 0, 0, 0)	22/30	2012.43	141.59	1919.28	2257.53	934.5
(0.5, 0.5, 0, 0, 0)	22/30	2016.57	138.44	1921.21	2245.28	1294.2
(0.6, 0.6, 0, 0, 0)	21/30	2027.38	143.96	1919.28	2248.08	1368.6
(0.7, 0.7, 0, 0, 0)	23/30	2009.21	132.73	1919.28	2248.08	1531.3
(0.75, 0.75, 0, 0, 0)	21/30	2025.96	143.54	1919.28	2245.28	1439.1
(0.8, 0.8, 0, 0, 0)	20/30	2038.75	147.56	1919.28	2248.08	1424.8
(0.833, 0.833, 0, 0, 0)	17/30	2067.71	154.65	1919.28	2248.08	1435.3
(0.9, 0.9, 0, 0, 0)	19/30	2049.26	152.23	1919.28	2260.63	1403.8

The results in Tables 5-7 illustrate the performance of the Weibull accepting GHC algorithm with the new neighborhood function. In particular, the results demonstrate differences when the values of K, M, and components of the probability switch vector are changed. The Weibull accepting GHC algorithm using the probability switch vectors (.1, .1, 0, 0, 0), (.2, .2, 0, 0, 0), (.3, .3, 0, 0, 0) with $\eta_2(s)$ resulted in a low probability of generating a manufacturing process design sequence other than s (thereby providing a myopic neighborhood structures), hence yielded inferior results than those obtained with the probability switch vectors containing higher probability components. This observation leads to the initial conclusion that as the number of manufacturing process design sequence changes increased, the mean value of the optimal solution found by the Weibull accepting algorithm improved.

The results in Tables 5-7 also suggest that choosing probability switch vectors that maximize process design sequence changes can also yield poor solutions, as measured by μ. This results from the algorithm moving too quickly out of optimal manufacturing process design sequences to non-optimal manufacturing process design sequences. Furthermore, the performance of a GHC algorithm depends on the values of K and M. Comparing the values for γ, μ, and σ in Tables 5 and 6, across the same probability switch vectors values in Tables 7, the Weibull accepting GHC algorithm with K = 50, M = 200 or K = 100, M = 100 yielded results that are significantly better than for the case with K = 200, M = 50 in Table 7. Moreover, the Weibull accepting GHC algorithm with K = 50, M = 200 or K = 100, M = 100 also increased the number of hill climbing solutions accepted during the execution of the algorithm. Examining (3), at each iteration of the Weibull accepting GHC algorithm, the probability that a hill climbing solution is accepted is large when the value of t_k is large. Therefore, using K = 50, M = 200, the temperature parameter, t_k, for the Weibull accepting GHC algorithm converges to zero at a sufficiently slow rate such that the probability of accepting a hill climbing solution also decreases very slowly at each outer loop iteration. This results in an increased frequency that the algorithm execution visits the optimal sequence at the beginning of the execution. Visiting the optimal sequence early in the execution of the algorithm also decreases the probability of becoming locked in a non-optimal process design sequence at the end of the algorithm execution.

Note that as a base case, to compare the effectiveness and value of the Weibull accepting algorithm with the new neighborhood function, experiments were run with a Weibull accepting algorithm using a neighborhood function (in place of η_2, termed the uniform neighborhood function) that assigns an integer value (i.e., 1,2,3,4,5) to each of the five possible manufacturing process design sequences, where the probability of moving from a design sequence to any other design sequence is.25. This algorithm was also executed using the same three combinations of values for K and M = M(k), k = 1,2,...,K. Thirty (independently seeded) replications of each GHC formulation were made. The mean (μ) and standard deviations (σ), as well as the minimum and maximum cost function values, were computed from the optimal values across these thirty replications. These results are given in Table 8.

TABLE 8
Weibull Accepting Algorithm Results with Uniform Neighborhood Function

Inner and Outer Loop Bounds	γ	μ	σ	Minimum	Maximum
K=50, M=200	24/30	2048.95	193.01	1919.28	2723.33
K=100, M=100	25/30	2147.12	623.76	1927.01	5200.51
K=200, M=50	19/30	2201.76	599.32	1924.94	5095.13

The results in Table 8, when compared to the results in Tables 5-7, suggest that using the Weibull accepting algorithm with the new neighborhood function can be more effective in identifying both the optimal design sequence and the optimal controllable input parameter values compared to the Weibull accepting algorithm using the uniform neighborhood function. Moreover, the variance of the optimal design sequences obtained using the new neighborhood function is significantly lower than for the uniform neighborhood function. These results can be explained by noting that the uniform neighborhood function selects neighboring design sequences uniformly, while the new neighborhood function weights this selection based, in part, on common manufacturing processes between the design sequences. This overlap provides a more effective strategy in moving between different design sequences, in search of the optimal design sequence.

The computational results are consistent with what would have been obtained using trial and error on the shop floor. In particular, the optimal design ($P_0P_2P_5P_6$) required smaller initial billets (hence there was less material wasted) and used machining processes (rather than forging and extrusion processes) to achieve the desired shape and size. These results are also consistent with those reported in Jacobson et al. (1998), in terms of the minimum cost solutions obtained. The advantage of using generalized hill climbing algorithms and computer simulation manufacturing processes is the speed and efficiency at which these results can be obtained, at a fraction of the cost that would be spent if trial and error on the shop floor would be required. Moreover, allowing a GHC algorithm to optimize over both the design sequences and the controllable input parameters provides an important first step to developing automated procedures for such optimization problems.

5. Simultaneous Generalized Hill Climbing Algorithms

This section introduces a mathematical framework for simultaneously addressing a set of related discrete optimization problems using local search algorithms. The resulting algorithms, termed *simultaneous generalized hill climbing* (SGHC) algorithms (Vaughan and Jacobson 2001), are motivated by the multiple sequence manufacturing process design optimization problems discussed in Section 4. SGHC algorithms can be applied to a wide variety of sets of related manufacturing, military and service industry discrete optimization problems. Many well-known local search algorithms can be embedded within the SGHC algorithm framework, including simulated annealing, threshold accepting, Monte Carlo search, and pure local search (among others).

A SGHC algorithm probabilistically moves between the set of related discrete optimization problems during its execution according to a problem generation probability function. The problem generation probability function is shown to be a stochastic process that satisfies the Markov property. Therefore, given a SGHC algorithm, movement between discrete optimization problems can be modeled as a Markov chain. Sufficient conditions that guarantee that this Markov chain has a uniform stationary probability distribution are obtained. Computational results are presented with SGHC algorithms for a set of traveling salesman problems. For comparison purposes, GHC algorithms are also applied individually to each traveling salesman problem. These computational results suggest that optimal/near optimal solutions can be reached more effectively using a SGHC algorithm.

It is common to encounter several discrete optimization problems where a relationship exists between the solution spaces of the individual problems. In general, these problems are approached individually. However, because of their similarities, the same computational tools can be effectively used to address them simultaneously. For example, the Material Process Design Branch of the Air Force Research Laboratory, Wright Patterson Air Force Base, has several similar discrete manufacturing process design optimization problems under study (see Jacobson et al. 1998, Sullivan and Jacobson 2000). These problems are difficult to solve due in part to the large number of design sequences and associated input parameter setting combinations that exist. Local search algorithms in the *generalized hill climbing* (GHC) algorithm framework were introduced to address such manufacturing design problems (Jacobson et al. 1998). Initial results with GHC algorithms required the manufacturing process design sequence to be fixed, with the GHC algorithm used to identify optimal input parameter settings for each feasible design sequence (Jacobson et al. 1998).

The SGHC algorithm framework is motivated by a study reported in Vaughan et al. (2000) which develops a new neighborhood function that allows a local search algorithm (in the GHC algorithm framework) to be used to also identify the optimal discrete manufacturing process design sequence among a set of valid design sequences. This neighborhood function allows the GHC algorithm to simultaneously optimize over both the design sequences and the input parameters, eliminating the need to approach each design sequence (i.e., a discrete optimization problem) individually. The computational results in Vaughan et al. (2000) suggest that such an approach is feasible and yields reasonable results. However the neighborhood function developed for this purpose is overly complex and problem specific.

The following analysis generalizes the results reported in Vaughan et al. (2000) by formally defining a class of sets of discrete optimization problems where a relationship exists, similar to the one described for the manufacturing problem. A set of discrete optimization problems contained in this class is defined as a set of *fundamentally related* discrete optimization problems. The SGHC algorithm framework is used to simultaneously approach sets of fundamentally related discrete optimization problems using GHC algorithms. A metric between elements in a set of fundamentally related discrete optimization problems is introduced to evaluate if and when it is advantageous to address a particular set of discrete optimization problems with a SGHC algorithm.

Vaughan et al. (2000) introduces a new neighborhood function for simultaneously addressing a set of related manufacturing process design optimization problems using GHC algorithms. This neighborhood function allows for simultaneous optimization across the design sequences and the controllable input parameters. The application of such optimization algorithms (that simultaneously explore multiple manufacturing process designs) using computer simulation is a new advance in how optimal manufacturing process designs can be efficiently identified (Vaughan et al. 2000).

It is common to encounter several discrete optimization problems where a relationship between the solution spaces of the individual problems exists. For example, Henderson et al. (2001) introduces a multiple platform search and rescue optimization problem that is modeled as several discrete optimization problems. They show that for this set, the solution spaces of the individual problems overlap and are a set of fundamentally related discrete optimization problems. Vaughan et al. (2000) describes an integrated blade rotor discrete manufacturing process design problem that illustrates how certain manufacturing problems can be modeled as several discrete optimization problems with overlapping solution spaces that satisfy the definition of fundamentally related discrete optimization problems. Note that this paper relaxes the methodology used to address the integrated blade rotor discrete manufacturing process design problem described in Vaughan et al. (2000) to develop a general mathematical framework for simultaneously approaching sets of fundamentally related discrete optimization problems.

To discuss the class of sets of discrete optimization problems for which SGHC algorithms are applicable, the following definitions are needed. Consider a *set of discrete optimization problems* $S = \{D_1, D_2, \ldots, D_m\}$, where each discrete optimization problem $D_y = (\Omega_y, f_y)$ is fully defined by a finite set of solutions Ω_y and a real-valued objective function $f_y: \Omega_y \to R$. A set of discrete optimization problems S is *fundamentally related* by a set $Ob = \{c_1, c_2, \ldots, c_n\}$ of objects if the solution space Ω_y of each discrete optimization problem $D_y = (\Omega_y, f_y) \in S$ can be fully defined by exactly one subset of *Ob*. Then for every discrete optimization problem $D_y = (\Omega_y, f_y) \in S$, there is exactly one set $C_y \subseteq Ob$ such that C_y completely defines Ω_y. The set C_y is defined to be the *fundamental relation set* of D_y.

Let S be a set of fundamentally related discrete optimization problems related by *Ob*. Consider $D_y \in S$ where $C_y \subseteq$ *Ob* is the fundamental relation set of D_y. Then C_y can be represented by the *binary activity vector* $\mathbf{c}^y \in \{0, 1\}^n$, $\mathbf{c}^y = (B_1, B_2, \ldots, B_n)$, where

$$B_i = \begin{cases} 1, & \text{if } c_i \text{ is contained in } C_y \\ 0, & \text{otherwise} \end{cases}.$$

SGHC algorithms are developed to approach sets of fundamentally related discrete optimization problems. When two discrete optimization problems, D_y and D_q, are contained in a set of fundamentally related discrete optimization problems with respective fundamental relation sets, C_y and C_q, where $|C_y \cap C_q| / |Ob|$ is close to one, it is reasonable to conjecture that the optimal/near optimal solutions of D_y and D_q are similar. The following detachment metric is designed to determine if two discrete optimization problems, in a set of fundamentally related discrete optimization, are close together, hence have similar solution spaces.

Let S be a set of fundamentally related discrete optimization problems related by *Ob*. To formally define the detachment metric ρ between discrete optimization problems $D_y, D_q \in S$, consider the metric space $<\Sigma^n, \rho>$, with $\Sigma = \{0, 1\}$, where the detachment metric ρ is defined on $\Sigma^n \times \Sigma^n$ such that the *distance* between two discrete optimization problems can be determined by considering their binary activity vectors $\mathbf{c}^y=(c_1^y, c_2^y, \ldots, c_n^y) \in \Sigma^n$ and $\mathbf{c}^q=(c_1^q, c_2^q, \ldots, c_n^q) \in \Sigma^n$. Define the *detachment metric* as

$$\rho(D_y, D_q) = \left|c_1^y - c_1^q\right| + \left|c_2^y - c_2^q\right| + \ldots + \left|c_n^y - c_n^q\right|$$

(Royden 1988, p.140). The detachment metric provides a way to measure the overlap (or lack of overlap) between the members in a set of fundamentally related discrete optimization problems.

To apply SGHC algorithms, a neighborhood function with an associated problem generation probability function for moving between discrete optimization problems during an execution of a SGHC algorithm must be developed. The neighborhood function is defined such that each discrete optimization problem has the entire set of discrete optimization problems as neighbors. Therefore, whenever this neighborhood function is applied, every discrete optimization problem is a *candidate problem* (i.e., has a positive probability of being selected). The *problem generation probability function* determines the probability of selecting a candidate problem.

More formally, define the neighborhood function, $\eta_{set}: S \rightarrow 2^S$, such that $\eta_{set}(D_y) = S$, for all $D_y \in S$. Define the *problem generation probability function* $h_{D_y D_q}(k, \rho(D_y, D_q))$, such that

$$0 < h_{D_y D_q}(k, \rho(D_y, D_q)) < 1, \text{ for every } D_y \in S,\ D_q \in \eta_{set}(D_y),$$

where

$$\sum_{D_q \in \eta_{set}(D_y)} h_{D_y D_q}(k, \rho(D_y, D_q)) = 1, \text{ for every } D_y \in S, D_q \in \eta_{set}(D_y)$$

for every k=1, 2, …, K.

Note that the problem generation probability function (the probability of selecting a candidate problem, $D_q \in \eta_{set}(D_y)$, $D_y \in S$) can be a function of both the outer loop iteration k=1, 2, …, K and the detachment metric $\rho(D_y, D_q)$.

SGHC algorithms provide a mathematical framework for addressing several fundamentally related discrete optimization problems simultaneously using GHC algorithms. SGHC algorithms seek to find optimal solutions for sets of fundamentally related discrete optimization problems by allowing the algorithm to probabilistically move between discrete optimization problems. When a new discrete optimization problem is generated, an initial solution for this new problem is also generated using information from the previous discrete optimization problem's final solution. The inner and outer loop structure defined for GHC algorithms can be used in SGHC algorithms, where SGHC algorithms restrict possible movement between discrete optimization problems to the first iteration of the outer loop iterations. This constraint ensures that a GHC algorithm is applied to each discrete optimization problem at least N(k) iterations each time it is generated (i.e., initially visited). Note that this was not the case for the manufacturing problem presented in Vaughan et al. (2000), where movement between discrete optimization problems was possible during all inner loop iterations. The SGHC algorithm is presented below in pseudo-code form.

```
Set the outer loop counter bound K and the inner loop counter bounds $N(k)$, $k=0,1,2,\ldots,K$
Define a set of hill climbing (random) variables $R_k: \Omega \times \Omega \rightarrow \Re \cup \{-\infty,+\infty\}$, $k=1,2,\ldots,K$
Set the iteration indices $N(0)=i=0$, $k=n=1$
Select an initial discrete optimization problem $D(0) \in S$
Select an initial solution $\omega(0,0) \in \Omega(0)$
Repeat while $k \leq K$
    Generate a discrete optimization problem $D(k) \in \eta_{set}(D(k-1))$
    If $D(k) \neq D(k-1)$, generate a solution $\omega \in \Omega(k)$ and $\omega(k, 1) \leftarrow \omega$
    else $\omega(k, 1) \leftarrow \omega(k-1, N(k-1))$
    Repeat while $n \leq N(k)$
        Generate a solution $\omega \in \eta(\omega(k, i))$
        Compute $\delta(\omega(k, i),\omega) = f(\omega)-f(\omega(k, i))$
        If $R_k(\omega(k, i),\omega) \geq \delta(\omega(k, i),\omega)$, then $\omega(k, i+1) \leftarrow \omega$
        If $R_k(\omega(k, i),\omega) < \delta(\omega(k, i),\omega)$, then $\omega(k, i+1) \leftarrow \omega(k, i)$
        $n \leftarrow n+1$, $i \leftarrow i+1$
    Until $n = N(k)$
    $n \leftarrow 1$, $k \leftarrow k+1$
Until $k = K$
```

All SGHC algorithms are formulated using three components, a set of hill climbing random variables $\{R_k\}$, a neighborhood function η between solutions, and a neighborhood function η_{set} between discrete optimization problems. The two-tuple (k,) represents the inner loop iteration i=1, 2, ..., N(k), during outer loop iteration k=1, 2, ..., K. D(k) is the discrete optimization problem the algorithm is executing over during the kth outer loop iteration, k=1, 2, ..., K, where the solution space of D(k) is depicted by $\Omega(k)$.

Markov chain theory is an effective tool for studying the performance of local search algorithms. The following analysis shows that an application of the SGHC algorithm can be modeled using Markov chains. In particular, an application of the GHC algorithm is first modeled using a Markov chain. Then an application of the SGHC algorithm is modeled by a set of Markov Chains.

To show that an application of the GHC algorithm can be modeled with a Markov chain, the following definitions are needed. A *stochastic process* is a family of random variables defined on some state space. If there are countable many members of the family, the process (termed a *discrete-time process*) is denoted by $Q_1, Q_2, \ldots$, where the set of distinct values assumed by a stochastic process is the *state space.* If the state space is countable or finite, the process is a *chain.* A stochastic process $\{Q_k\}$, k=1,2, ... with state space $\Omega = \{\omega_1, \omega_2, \ldots\}$ satisfies the *Markov property* if for every n and for all states $\omega_1, \omega_2, \ldots, \omega_n$

$$\Pr\{Q_n=\omega_n \mid Q_{n-1}=\omega_{n-1}, Q_{n-2}=\omega_{n-2}, \ldots, Q_1=\omega_1\} = \Pr\{Q_n=\omega_n \mid Q_{n-1}=\omega_{n-1}\} = P_{n(n-1)}.$$

A discrete-time stochastic process that satisfies the Markov property is a *Markov chain* (Isaacson and Madsen 1985).

Let $\{Q_k\}$ denote a discrete-time Markov chain with finite solution space $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$. For this chain there are $|\Omega|^2$ transition probabilities, $\{P_{ij}\}$, i,j = 1, 2, ..., $|\Omega|$. The *transition matrix* associated with the Markov chain $\{Q_k\}$ is P, where P_{ij} is the probability of moving from state ω_i to state ω_j.

An application of a GHC algorithm can be modeled by a stochastic process $\{Q_n^k\}$, k = 1, 2, ..., K, n = 1, 2, ..., N(k), $Q_n^k \in \Omega$ with solution space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ that satisfies the Markov property for every n and all states $\omega_1, \omega_2, \ldots, \omega_n$ (i.e., $\{Q_n^k\}$ is a Markov chain). To see this, consider an application of a GHC algorithm to a discrete optimization problem with solution space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$. Define $g_{ij}(k)$ to be the *generation probability function* for the neighborhood function η, where $g_{ij}(k)$ is the probability that $\omega_j \in \eta(\omega_i)$ is generated during outer loop iteration k. Consider the inner loop iterations for fixed outer loop iteration k = 1,2,...,K. Let $\{Q_n^k\}$, k =1,2,...,K, n = 1,2,...,N(k), $Q_n^k \in \Omega$, be the stochastic process where if $Q_n^k = \omega_i$, then the GHC algorithm is at solution ω_i during inner loop iteration n and outer loop iteration k (Johnson and Jacobson 2001b). If the GHC algorithm is at solution ω_i at inner loop iteration n-1, the probability that the algorithm is at solution ω_j at inner loop iteration n is

$$P_{ij}(k) = \begin{cases} g_{ij}(k)\Pr(R_k(\omega_i,\omega_j)) \geq \delta_{ij} & \text{for all } \omega_i \in \Omega, \omega_j \in \eta(\omega_i), j \neq i \\ 1 - \sum\limits_{\substack{z\in\eta(\omega_i)\\ z\neq i}} P_{iz}(k) & j = i \\ 0 & \text{otherwise} \end{cases},$$

independent of the solutions the algorithm visited at inner loop iterations 1, 2, …, n-2. Therefore

$$\Pr\{Q_n^k=\omega_j \mid Q_{n-1}^k=\omega_i,\ Q_{n-2}^k=\omega_{i\,(n-2)}, \ldots, Q_1^k=\omega_{i\,(1)}\} = \Pr\{Q_n^k=\omega_j \mid Q_{n-1}^k=\omega_i\} = P_{ij}(k),$$

Hence the Markov property holds. Moreover, for every outer loop iteration k, the Markov chain $\{Q_n^k\}$ has a transition matrix P(k), where $P_{ij}(k)$ is as defined above .

Recall, that a SGHC algorithm is applied to a set of fundamentally related discrete optimization problems. Movement between discrete optimization problems is only possible at outer loop iterations k=1, 2, …, K. During the inner loop iterations, the SGHC algorithm is executing over the solution space of the current discrete optimization problem using a GHC algorithm

The following analysis shows that for fixed outer loop iteration k = 1, 2, …, K, the stochastic process corresponding to the SGHC algorithm solution at inner loop iterations n = 1, 2, …, N(k) can be modeled by a Markov chain that corresponds to an application of a GHC algorithm. Moreover, it is shown that for outer loop iterations k = 1, 2, …, K, the possible movement between discrete optimization problems is a stochastic process that satisfies the Markov property.

Consider an application of a SGHC algorithm to a set of fundamentally related discrete optimization problems S = {D_1, D_2, …, D_m}, where each discrete optimization problem D_y, y = 1, 2, …, m is fully defined by a solution space Ω_y and an objective function f_y (i.e., $D_y=(\Omega_y, f_y)$). Consider the inner loop iterations n = 1, 2, …, N(k), for fixed outer loop iteration k, k = 1, 2, …, K. Let $\{Q_n^k(D_y)\}$, k = 1, 2, …, K, n = 1, 2, …, N(k) be the stochastic process where if $Q_n^k(D_y) = \omega_i$, then the SGHC algorithm is at solution $\omega_i \in \Omega_y$ at inner loop iteration n of outer loop iteration k.

Note that, for all inner loop iterations n = 1, 2, …, N(k) of an outer loop iteration k = 1, 2, …, K, the SGHC algorithm is executing over a particular discrete optimization problem from the set of fundamentally related discrete optimization problems S={D_1, D_2, ..., D_m} using a GHC algorithm. It was shown that any application of a GHC algorithm to a discrete optimization problem can be modeled as a Markov chain. Therefore, for fixed outer loop iteration k, the stochastic processes $\{Q_n^k(D_y)\}$, y = 1, 2, …, m, with transition matrices P_y, are the Markov chains that correspond to an application of the GHC algorithm to the discrete optimization problems D_y, y = 1, 2, ..., m for every n = 1, 2, ..., N(k) and for all states $\omega_1, \omega_2, \ldots, \omega_{|\Omega_y|}$.

Movement between discrete optimization problems is a stochastic process that satisfies the Markov property. To see this, define $\{\Psi(k)\}$, $\Psi(k) \in S$, k = 1, 2, ... to be the stochastic process where if $\Psi(k)=y$, then during outer loop iteration k, for all inner loop iterations n = 1, 2, ..., N(k) the SGHC algorithm is executing over solutions contained in the solution space of the discrete optimization problem $D_y = (\Omega_y, f_y)$. If the SGHC algorithm is executing over Ω_y at outer loop iteration k-1, then the probability that the SGHC algorithm is executing over Ω_q during outer loop iteration k is

$$T_{yq}(k) = h_{D_yD_q}(k, \rho(D_y, D_q)),$$

independent of the discrete optimization problems the SGHC algorithm visited at outer loop iterations 1, 2, ..., k-2 and independent of all preceding inner loop iterations. Therefore,

$$\Pr\{\Psi(k)=q \mid \Psi(k-1)=y, \Psi(k-2)=y_{k-2}, \ldots, \Psi(1)=y_1\} = \Pr\{\Psi(k)=q \mid \Psi(k-1)=y\} = T_{yq}(k)$$

and the Markov property holds. Moreover, the Markov chain $\{\Psi(k)\}$ has transition matrix T(k), where $T_{yq}(k)$ is as defined above.

Consider an application of the SGHC algorithm to a set of fundamentally related discrete optimization problems, S. This section presents sufficient conditions that guarantee that a SGHC algorithm will (as k approaches $+\infty$) be executing over the solution space of each discrete optimization problem $D_y \in S$ with probability 1/ |S|, where |S| is the cardinality of S. This result implies that, as k approaches $+\infty$, each discrete optimization problem in $S = \{D_1, D_2, \ldots, D_m\}$ is being explored with equal probability.

This section develops sufficient conditions that place restrictions on the selection of the problem generation probability function $h_{D_yD_q}(k, \rho(D_y, D_q))$. A discrete time Markov chain is a *stationary Markov chain* if the probability of going from one state to another state is independent of the iteration at which the transition is being made (Isaacson and Madsen 1985). That is, let $\{X_n\}$ be a stationary Markov chain with state space $S=\{D_1, D_2, \ldots, D_m\}$. Then for all states D_y and D_q, for all k = -(n-1),-(n-2), ...,-1, 0, 1, 2, ... , $\Pr\{X_n=D_y \mid X_{n-1}=D_q\} = \Pr\{X_{n+k}=D_y \mid X_{n+k-1}=D_q\}$.

The *long run distribution (stationary probability distribution)* of a stationary Markov chain with corresponding transition matrix T is defined by $\pi = [\pi_1 \quad \pi_2 \quad \cdots \quad \pi_m]$, $\pi_i \geq 0$, for all i = 1, 2, ..., m, where $\pi = \pi T$ and $\sum_{i=1}^{m} \pi_i = 1$. Equivalently, the long run distribution of a stationary Markov chain is defined by $\pi = [\pi_1 \quad \pi_2 \quad \cdots \quad \pi_m]$, where

$$\pi_j = \lim_{n \to +\infty} T_{ij}^{(n)}.$$

If Markov chain $\{\Psi(k)\}$ is stationary and has a uniform long run distribution, then as k approaches infinity the SGHC algorithm is executing over the solution space of each discrete optimization problem in $S = \{D_1, D_2, \ldots, D_m\}$ with probability $1 / m = 1 / |S|$. Theorem 2 provides sufficient conditions for the selection of the problem generation probability function $h_{D_yD_q}(k, \rho(D_y, D_q))$ that guarantee that the Markov chain $\{\Psi(k)\}$ has a uniform long run distribution. Therefore, when the sufficient conditions of Theorem 2 hold, the SGHC algorithm will (as k approaches $+\infty$) be in discrete optimization problem $D_y \in S$, $y = 1, 2, \ldots, m$ with probability $1 / |S|$.

To prove Theorem 2, the following definitions are needed. A subset, C, of the state space, S, is *closed* if $P_{ij} = 0$ for all $i \in C$, $j \notin C$. A Markov chain is *irreducible* if there exists no nonempty closed set other than S itself. If S has a proper closed subset, it is *reducible*. State ω_i is said to have *period d* if $P_{ii}^{n} = 0$ whenever n is not divisible by d and d is the greatest integer with this property. A state with period one is said to be *aperiodic* (Isaacson and Madsen 1985).

Theorem 2 (Vaughan and Jacobson 2001): Consider an application of the SGHC algorithm. Define $h_{D_yD_q}(k, \rho(D_y, D_q)) = h_{D_yD_q}(\rho(D_y, D_q)) = h_{D_qD_y}(\rho(D_y, D_q))$, for every $k \in Z^+$. Consider the transition matrix T defined by $T_{yq} = h_{D_yD_q}(\rho(D_y, D_q))$. If the transition matrix T is irreducible and aperiodic, then the Markov chain $\{\Psi(k)\}$ has a uniform long run distribution. Moreover, the long run distribution of $\{\Psi(k)\}$ is $\pi = [1/|S|\ 1/|S| \ldots 1/|S|]$.

Theorem 2 shows that if the problem generation probability function is chosen such that the Markov chain $\{\Psi(k)\}$ is stationary and the associated transition matrix is symmetric, then the Markov chain $\{\Psi(k)\}$ has a uniform stationary distribution.

The following discussion provides an illustrative example of a set of fundamentally related discrete optimization problems. The Traveling Salesman Problem is formally described. Then the Multiple Traveling Salesman Problem is presented and formulated as a set of fundamentally related discrete optimization problems.

The traveling salesman problem (TSP) is a well-known NP-hard discrete optimization problem (Lawler et al. 1985). The TSP is used to illustrate various local search algorithms because it is useful for modeling a variety of real world problems. For example, traditional applications of the TSP include a variety of vehicle routing and scheduling problems. More recently, applications of the TSP have been expanded to include modern applications like the printing of circuit boards, x-ray crystallography, overhauling of gas turbine engines, and the controlling of industrial robots (Johnson and Jacobson 2001b).

To formally define the TSP the following definitions are needed (Lawler et al. 1985). Define a *graph* to be a finite set of vertices, some pairs of which are joined by edges. A *cycle* in a graph is a set of vertices of the graph, which is such that it is possible to move from vertex to vertex, along edges of the graph, so that all vertices are encountered exactly once, finishing at the start. If a cycle contains all the vertices of the graph, it is called a *Hamiltonian cycle (or tour)*. The TSP is defined as follows (Garey and Johnson 1979).

Traveling Salesman Problem (TSP)
Instance: Given a set of n cities $C = \{c_1, c_2, \ldots, c_n\}$ and a distance matrix D that represents the cost of traveling between the cities in the set C.

Question: Find a Hamiltonian cycle $H = (c_1, c_2, \ldots, c_n)$ such that $f(H) = \sum_{j=1}^{n-1} D(c_j, c_{j+1}) + D(c_n, c_1)$ is minimized.

An instance of a TSP is a discrete optimization problem, where the solution space is the set of possible all Hamiltonian cycles (with each tour consisting of n cities), $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{\frac{(n-1)!}{2}}\}$. The objective function value for each solution $\omega_i=(c_1, c_2, \ldots, c_n) \in \Omega$ is the sum of the distances the tour depicts, $f(\omega_i) = \sum_{j=1}^{n-1} D(c_j, c_{j+1}) + D(c_n, c_1)$. The optimal objective function value represents the shortest distance traveled.

The Multiple Traveling Salesman Problem (MTSP) is introduced to illustrate the application of SGHC algorithms. The MTSP is defined as follows.

Multiple Traveling Salesman Problem (MTSP)
Instance: Given a set of n cities $Ob = \{c_1, c_2, \ldots, c_n\}$, a set of m subsets of Ob, $O = \{C_1, C_2, \ldots, C_m\}$, and a distance matrix D that represents the cost of traveling between the cities in the set Ob.
Question: Find a Hamiltonian cycle $H = (c_1, c_2, \ldots, c_{|\Omega_y|})$ where there exists a C_y, $y = 1, 2, \ldots, m$ such that $c_j \in C_y$, for every $j = 1, 2, \ldots, n$, and $f(H) = \sum_{j=1}^{n-1} D(c_j, c_{j+1}) + D(c_n, c_1)$ is minimized.

Note that each of the sets $C_y \in O$ represents an instance of the TSP, D_y. Since the TSP can be formulated as a discrete optimization problem, then the MTSP problem can be represented by set of discrete optimization problems $S=\{D_1, D_2, \ldots, D_m\}$. The set $S = \{D_1, D_2, \ldots, D_m\}$ is a set of fundamentally related discrete optimization problems. To see this, note that each discrete optimization problem $D_y \in S$, $y = 1, 2, \ldots, m$ is fully defined by $C_y \subseteq Ob = \{c_1, c_2, \ldots, c_n\}$. Therefore, C_y is the fundamental relation set of discrete optimization problem D_y.

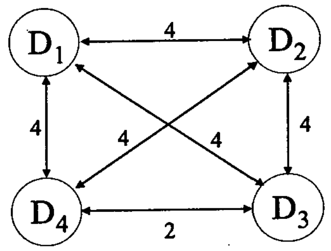
The following computational results illustrate how a MTSP can be addressed with SGHC algorithms. For comparison purposes, GHC algorithms are also applied to the individual members in the set of fundamentally related discrete optimization problems. These computational results suggest that optimal/near optimal solutions can be reached in less total iterations using a SGHC algorithm.

To develop a MTSP, a set consisting of 20 cities was generated by randomly locating each city on a 1000 by 1000 unit grid. Four TSPs of size 18 were generated by randomly selecting 18 of the 20 cities for each traveling salesman problem. In the case where an identical set of cities was generated for two or more of the problems, a completely new set of discrete optimization problems was generated, resulting in four distinct randomly generated TSPs.

The four randomly generated TSPs are arbitrarily labeled D_1, D_2, D_3, and D_4. The detachment metric $\rho(D_y, D_q)$, between the TSPs, was calculated for all y,q = 1, 2, 3, 4. The distance matrix and distance diagram are depicted in Figure 2.

Figure 2: Distance Matrix and Distance Diagram

	D_1	D_2	D_3	D_4
D_1	0	4	4	4
D_2	4	0	4	4
D_3	4	4	0	2
D_4	4	4	2	0



Computational results with Monte Carlo search, pure local search, and simulated annealing using SGHC algorithms are reported. For comparison purposes, computational results with Monte Carlo search, pure local search, and simulated annealing using GHC algorithms are also reported. The 2-Opt neighborhood function was used for all executions of the SGHC and GHC algorithms. For the SGHC algorithms, the problem generation probability function is defined as

$$h_{D_yD_q}(k, \rho(D_y, D_q)) = [1/\rho(D_y, D_q)] / [\sum_{\substack{i=1 \\ i\neq y}}^{4} (1/\rho(D_y, D_i))],\ y \neq q$$

and

$$h_{D_yD_y}(k, \rho(D_y, D_y)) = 1 - \sum_{\substack{q=1 \\ q\neq y}}^{4} [1/(\rho(D_y, D_q)] / [\sum_{\substack{i=1 \\ i\neq y}}^{4} (1/\rho(D_y, D_i))] = 0,$$

for every y,q = 1, 2, 3, 4, y ≠ q for every k = 1, 2, …, K.

This problem generation probability function is such that the associated transition matrix is symmetric and the Markov chain $\{\Psi(k)\}$ is stationary. Therefore, by Theorem 1, the Markov chain $\{\Psi(k)\}$ has a uniform stationary distribution, hence as k approaches infinity, the SGHC algorithm is executing over the solution space of each discrete optimization problem in $S = \{D_1, D_2, \ldots, D_m\}$ with probability 1/m =

1/|S| = 1/4. Moreover, this problem generation function guarantees the discrete optimization problem over which the SGHC algorithm is executing changes at every outer loop iteration k (i.e., $\Psi(k) \neq \Psi(k-1)$, for all $k = 1, 2, \dots$).

Executions with different values of K and N=N(k), k = 1, 2, …, K are reported. To compare the performance of applying a SGHC algorithm versus applying a GHC algorithm, the inner and outer loop bounds of the SGHC algorithm were doubled. Therefore, the total number of iterations that the SGHC algorithm executes is equal to the total number of iterations executed using the GHC algorithm for the four individual problems. Let $R \in Z^+$ represent the total number of replications executed for each SGHC and GHC algorithm formulation. For each replication, a different randomly generated initial tour was used. The means, μ, standard deviations, σ, and the minimum and maximum distances, were computed from the optimal tour distances across these R replications. The value γ in Tables 9 to 14 represents the number of replications for which the algorithms find the minimum distance tour. For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 (i.e., $t_k=\beta t_{k-1}$), with initial temperature parameter t_0=2000.

Table 9: GHC Algorithm Results: Pure Local Search

Outer and Inner Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	4/30	3864.3	36.9726	3805.4	3916.0
K=200, M=100	3/30	3863.4	32.7691	3805.4	3907.3
K=300, M=75	1/30	3876.6	36.9549	3805.4	3953.7
K=400, M=75	1/30	3885.3	42.1893	3805.4	3973.4
K=400, M=50	2/30	3867.9	37.7520	3805.4	3916.0
K=800, M=50	1/15	3872.8	35.2469	3805.4	3916.0

Table 10: SGHC Algorithm Results: Pure Local Search

Outer and Inner Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	10/30	3819.4	13.2943	3805.4	3831.8
K=200, M=100	23/30	3808.9	9.0535	3805.4	3831.6
K=300, M=75	23/30	3808.9	9.0535	3805.4	3831.6
K=400, M=75	24/30	3807.2	6.6434	3805.4	3831.6
K=400, M=50	9/30	3818.5	13.3165	3805.4	3831.6
K=800, M=50	12/15	3810.7	10.8417	3805.4	3831.6

Table 9 suggests that when the number of outer loop iterations for a pure local search GHC algorithm is increased from 100 to 200, performance of the algorithm shows no improvement.

However, Table 10 suggests that the performance of a pure local search SGHC algorithm improves significantly (as measured by μ) when the number of outer loop iterations is increased from 100 to 200.

Table 11: GHC Algorithm Results: Simulated Annealing

Outer and Inner Loop Bounds	t_0	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	3000	1/30	3854.1	41.4192	3805.4	3953.7
K=200, M=100	3000	1/30	3861.2	35.7662	3805.4	3916.0
K=300, M=75	2000	2/30	3861.5	39.7074	3805.4	3973.4
K=400, M=75	2000	5/30	3855.9	35.3872	3805.4	3907.5
K=400, M=50	2000	3/30	3868.8	39.4863	3805.4	3916.0
K=800, M=50	2000	1/15	3885.9	29.7020	3805.4	3916.0

Table 12: SGHC Algorithm Results: Simulated Annealing

Outer and Inner Loop Bounds	t_0	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	3000	16/30	3814.1	12.5549	3805.4	3831.6
K=200, M=100	3000	19/30	3808.0	7.9899	3805.4	3831.6
K=300, M=75	2000	20/30	3808.9	9.0535	3805.4	3831.6
K=400, M=75	2000	23/30	3808.9	9.0535	3805.4	3831.6
K=400, M=50	2000	7/30	3831.6	13.1248	3805.4	3831.6
K=800, M=50	2000	1/15	3813.6	12.5352	3805.4	3831.6

Table 13: GHC Algorithm Results: Monte Carlo Search

Outer and Inner Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	1/30	6390.0	294.6998	5634.9	6805.9
K=200, M=100	1/30	6195.6	239.7325	5575.7	6656.3
K=300, M=75	1/30	6111.7	293.6761	5293.0	6613.9
K=400, M=75	1/30	6099.6	292.5790	5310.2	6488.3
K=400, M=50	1/30	6122.6	287.3693	5291.7	6575.7
K=800, M=50	1/15	6007.1	231.746	5479.1	6328.1

Table 14: SGHC Algorithm Results: Monte Carlo Search

Outer and Inner Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	1/30	6758.2	202.9184	5793.3	6758.2
K=200, M=100	1/30	6109.3	301.5243	5243.7	6611.5
K=300, M=75	1/30	6214.6	204.3092	5727.6	6575.3
K=400, M=75	1/30	6054.3	232.6718	5614.9	6462.8
K=400, M=50	1/30	6265.8	197.6956	5877.3	6659.3
K=800, M=50	1/15	5954.7	301.7151	5299.6	6382.3

Tables 13 and 14 suggest that there is little difference in performance between Monte Carlo search GHC algorithms and Monte Carlo search SGHC algorithms. Note that this result occurs since Monte Carlo search accepts neighboring solutions independent of their objective function value. Therefore, when movement between discrete optimization problems occurs there is no exchange of common information between the discrete optimization problems. Overall, Tables 9 through 12 suggest that the SGHC algorithms outperform the GHC algorithms. The minimum distance found over the R replications using SGHC algorithms is significantly smaller than the minimum distance found over the R replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Additionally, the standard deviation of the optimal values over the R replications is smaller using the SGHC algorithms.

SGHC algorithms provide a new approach for addressing a set of fundamentally related discrete optimization problems that can be more efficient than the traditional approach of addressing each discrete optimization problem in the set S individually with a local search algorithm. For example, SGHC algorithms allow practitioners to make a single algorithm run over a set of fundamentally related discrete optimization problems. Moreover, the computational results presented suggest that a SGHC algorithm can outperform the GHC algorithm. The development of the SGHC algorithm and the mathematical results in this paper make it possible for the SGHC algorithm to be adapted and used to approach a variety of real world problems.

Several on-site meetings and interactions with Austral Engineering and Software, Inc., has resulted in further enhancement to the generalized hill climbing algorithm framework and software that our research group has developed for their use through this project. This on-going effort of developing and enhancing the generalized hill climbing algorithm code, in conjunction with the software development efforts of Austral Engineering and Software, Inc., as well as the theoretical work being undertaken to better understand the generalized hill climbing algorithm framework and how to exploit its form to improve performance, all serve as a powerful mechanism for transitioning this research from an academic environment into an industrial setting.

REFERENCES

Chen, C.H., Kumar, V., 1996, "Motion Planning of Walking Robots in Environments with Uncertainty," *Proceedings of IEEE Conference on Robotics and Automation.*

Dai, L., 1996, "Convergence Properties of Ordinal Comparison in the Simulation of Discrete Event Dynamic Systems, " *Journal of Optimization Theory and Application*, 91(2), 363-388.

Fischer, C.E., Gunasekera, J.S., Malas, J.C., 1997, "Process Model Development for Optimization of Forged Disk Manufacturing Processes," *Steel Forgings*, Second Volume, ASTM STP 1257, E.G. Nisbett and A.S. Melilli, Editors, American Society for Testing and Materials.

Garey, M.R., Johnson, D.S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, New York.

Gunasekera, J.S., Fischer, C.E., Malas, J.C., Mullins, W.M., Yang, M.S., 1996, "Development of Process Models for Use with Global Optimization of a Manufacturing System," *Proceedings of the ASME Symposium on Modeling, Simulation and Control of Metal Processing*, ASME-International Mechanical Engineering Congress, Atlanta, GA, November, 1996.

Henderson, D., Vaughan, D.E., Jacobson, S.H., Wakefield, R.R., 2001, " Optimal Search Strategies Using Simultaneous Generalized Hill Climbing Algorithms." (Barchi Prize Nomination)

Ho, Y.C., Sreenivas, R., Vakili, P., 1992, "Ordinal Optimization of Discrete Event Dynamic Systems," *Journal of Discrete Event Dynamical Systems,* 2(2), 61-88.

Isaacson, D., Madsen, R., 1985, *Markov Chains Theory and Applications*, Robert E. Krieger Publishing Company, Inc., Malabar, Florida.

Jacobson, S.H., Sullivan, K.A., Johnson, A.W., 1998, "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms," *Engineering Optimization*, 31, 247-260.

Johnson, A.W., 1996, "Generalized Hill Climbing Algorithms for Discrete Optimization Problems," Ph.D. Dissertation, Department of Industrial and Systems Eng., Virginia Tech, Blacksburg, Virginia.

Johnson, A.W., Jacobson, S.H., 2001a, "A Convergence Result for a Class of Generalized Hill Climbing Algorithms," *Applied Mathematics and Computation* (Accepted).

Johnson, A.W., Jacobson, S.H., 2001b, "A General Convergence Result for Hill Climbing Algorithms," *Discrete Applied Mathematics* (Accepted).

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., 1985, *The Traveling Salesman Problem*, John Wiley and Sons, Chichester.

Lee, L.H, Lau, T.W.E., Ho, Y.C. 1998, "Explanation of Goal Softening in Ordinal Optimization," *IEEE Transactions on Automatic Control.*

Reeves, C.R., 1993, *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, Inc., New York, New York.

Royden, H.L., 1988, *Real Analysis*, Third Edition, Prentice Hall, New Jersey.

Sullivan, K.A., 1999, "A Convergence Analysis of Generalized Hill Climbing Algorithms," Ph.D. Dissertation, Department of Industrial & Systems Engineering, Virginia Tech, Blacksburg, Virginia.

Sullivan, K.A., Jacobson, S.H., 2000, "Ordinal Hill Climbing Algorithms for Discrete Manufacturing Process Design Optimization Problems," *Discrete Event Dynamical Systems* 10(4), 307-324.

Sullivan, K.A., Jacobson, S.H., 2001, "A Convergence Analysis of Generalized Hill Climbing Algorithms," *IEEE Transaction on Automatic Control* (Accepted).

Vaughan, D.E., Jacobson, S.H., 2001, "Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problems," Technical Report, University of Illinois, Urbana, Illinois.

Vaughan, D.E., Jacobson, S.H., Armstrong, D.E., 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," *ASME Journal of Mechanical Design*, 122(2), 164-171.

APPENDIX

The following are the input parameters for the seven processes. Note that all the length measurements (i.e., radius, height) are in inches, and all the temperatures are in fahrenheit. All speeds are in inches per second. The die friction factors are unitless.

Process	Input Parameters	Possible Values
P_0	Billet Radius	[2,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0]
	Billet Height	[1.0,2.0,3.0] * Billet Radius
P_1	Die Geometry:	
	Height 1	[1.5, 2.0, 2.5, 3.0,3.5]
	Height 2	[1.5, 2.0, 2.5, 3.0,3.5]
	Height 3	[1.5, 2.0, 2.5, 3.0,3.5]
	Die Speed	[.4,.6,.8,1.0,1.2]
	Die Friction Factor	[.2,.4,.6,.8]
	Die Temperature/Ambient Temperature	[1562,1607,1652,1697,1742,1787,1832]
P_2	Radius	[2.0,2.25,...,5.0]
	Height	[1.5,2.0,2.5,3.0,3.5]
P_3	Die Geometry:	
	Radius 1	[1.5, 1.75, ..., 3.5]
	Height 1	[0.5, 0.625, 0.75, 0.875]
	Radius 2	9.0 (fixed)
	Height 2	[1.0,1.25,1.5,1.75,2.0]
	Die Speed	[0.4,0.6,0.8,1.0,1.2]
	Die Friction Factor	[0.2,0.4,0.6,0.8]
	Die Temperature/Ambient Temperature	[1562,1607,1652,1697,1742,1787,1832]
P_4	Radius 1	[1.25, 1.50, ..., 3.25]
	Height 1	[0.5,0.625,0.75,0.875]
	Radius 2	[1.5, 1.75, ..., 3.5]
	Height 2	[0.5,0.75,1.0,1.25,1.5,1.75,2.0]
	Radius 3	[3.5, 3.75, ..., 5.0]
	Height 3	[1.0,1.25,1.5,1.75,2.0]
P_5	Radius 1	3.0 (fixed)
	Height 1	0.5 (fixed)
	Radius 2	3.0 (fixed)
	Height 2	1.5 (fixed)
	Radius 3	4.0 (fixed)
	Height 3	1.5 (fixed)
P_6	Radius	[1.5,2.0,....,4.0]
	Velocity	[0.1,0.5,1.0,1.5]
	Die Friction Factor	[0.2,0.4,0.6,0.8]
	Die Length	[4.0,6.0,8.0,10.0]
	Temperature	[1562,1652,.......,2372]

(* Reminder: there is a option for Die Geometry, shape =1 for conical shape, this value is fixed.)

MEDIA COVERAGE

The paper "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," (co-authored with Diane E. Vaughan and Derek E. Armstrong) published in *ASME Journal of Mechanical Design* (Volume 122(2), 164-171) was featured in an article on October 20, 2000 focusing on new directions in manufacturing research in *Advanced Manufacturing Technology* (www.wiley.com/technical_insights).

TRANSITIONS

Extensive interactions with Austral Engineering and Software (AES), Incorporated, have resulted in generalized hill climbing algorithm commercial quality software code. Mr. Armstrong spending ten weeks working at the Materials Process Design Branch, Wright Patterson Air Force Base in collaboration with AES, facilitated these interactions. The resulting code has been developed as library functions that are compatible with the interactive framework (MPDX©) being developed by Austral Engineering and Software, Incorporated, as part of their SBIR Phase II contract through Wright Patterson Air Force Base.

AWARDS

Dr. Diane Vaughan was awarded second place in the 2001 Paul E. Torgersen Graduate Student Research Excellence Award at Virginia Tech. This competition evaluates all the Ph.D. dissertations that were awarded within the College of Engineering at Virginia Tech over the prior twelve month period. See http://www.eng.vt.edu/eng/grad-students/PETAward/petaward.html for further details.

PUBLICATIONS

The following is a list of publications that have resulted from research on this grant.

Submitted but not yet Accepted

Henderson, D., Vaughan, D.E., Jacobson, S.H., Wakefield, R.R., Sewell, E.C. "Optimal Cut and Fill Strategies using Local Search Algorithms."

Henderson, D., Vaughan, D.E., Jacobson, S.H., Wakefield, R.R., " Optimal Search Strategies Using Simultaneous Generalized Hill Climbing Algorithms." (Military Operations Research Society Conference Barchi Prize Nomination)

Vaughan, D.E., Jacobson, S.H., "Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problems."

Published or Accepted

Jacobson, S.H., Sullivan, K.A., Johnson, A.W., 1998, "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms," *Engineering Optimization*, 31, 247-260.

Sullivan, K.A., Jacobson, S.H., 2000, "Ordinal Hill Climbing Algorithms for Discrete Manufacturing Process Design Optimization Problems," *Discrete Event Dynamical Systems*, 10(4), 307-324.

Sullivan, K.A., Jacobson, S.H., 2001, "A Convergence Analysis of Generalized Hill Climbing Algorithms," *IEEE Transactions on Automatic Control.*

Vaughan, D., Jacobson, S.H., Armstrong, D.E., 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," *ASME Journal of Mechanical Design*, 122(2), 164-171.